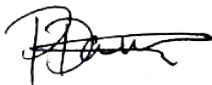




Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

School of Applied Computing

Student Name:	Peter Davies	Student P number:	P079868
Programme:	BSc Web Development		
Module:	Advanced Web Development	Lecturer:	Mike Dacey
Assignment Title:	Advanced Web Development Part One: Object Orientated PHP Web App Programming		
Hand Out Date:	December 2018	Submission deadline:	18/01/19

Disclosure:	I declare that this assignment is all my own work and that I have acknowledged all materials used from the published or unpublished works of other people. All references have been duly cited.		
Student's Signature:		Date:	19/01/19

NB. The assignment has been submitted late, however, lecturer, Mr. Michael Dacey has extended the deadline to Monday 21st January.

Contents

1. Introduction.....	1
2. Design Patterns.....	2
2.1 Design Patterns.....	2
2.2 Reasons for Using Design Patterns.....	5
2.3 Examples of Design Patterns in Web Development.....	6
2.3.1 Adapter Pattern.....	6
2.3.1.1 Adapter Pattern Example.....	7
2.3.2 Template Pattern.....	13
2.3.2.1 Template Pattern Example.....	13
3. Swansea City Archive Web Application.....	19
3.1 Description of Web Application.....	19
3.2 Git Version Control.....	19
3.3 Features of Web Application	20
3.3.1 Restricted User Views.....	20
3.3.2 Member Registration and Log-In.....	22
3.3.3 Player Searches.....	23
3.3.4 Pageable Fixtures.....	24
3.4 Design Patterns Used.....	26
3.4.1 Observer Pattern.....	26
3.4.1.1 Testing New User Observer.....	27
3.4.1.2 Testing Search Log Observer.....	29
3.4.2 Repository Pattern.....	30
3.4.3 Façade Pattern.....	34
4. Evaluation.....	38
5. References.....	39
6. Bibliography.....	41
7. Appendix.....	43

Introduction

The following report investigates design patterns used in advanced web development with PHP. It also documents the production of a web application using design patterns and object orientated programming techniques.

The report firstly looks at two design patterns: The *Adapter* pattern and the *Template* pattern and provides an explanation of each, with demonstrations of testing. Code for each pattern is supplied in the accompanying zip file.

Secondly, the report documents the production of a web application produced for fans of Swansea City Football Club using PHP OOP techniques and attempt at implementing design patterns. Each feature and pattern are discussed and a zip file containing all code accompanies this report along with a full code listing in the appendix which follows an evaluation, reference list and bibliography.

Design Patterns

The following section will look at what design patterns in programming/web development are, followed by a brief history of design patterns, why programmers use them, their use in real world scenarios and examples.

2.1 Design Patterns

Developers, in any profession will face problems which have been encountered by their peers before them. Alexander [1] states:

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice"

Alexander's statement refers to his architectural work for use of patterns when designing buildings and towns, however, this has become a statement adopted by developers when describing object-orientated programming pattern solutions, however, instead of conveying solutions with use of roofs or windows, developers convey solutions with objects and interfaces [2].

Meaning that design patterns are distillations of common problems encountered throughout the developing community: useful as they provide definitions of tested solutions and describe likely outcomes. They are the tried and tested best practice solution to a problem, found through results of good analysis the problem [3]. Or *"solutions to a problem in a context"* [2].

For this assignment, the developer will encounter the following problems: how can we separate data from an application? How can allow different users different access rights to a web application? How do we write logs of searches made or attempted log-ins to a web app to a database? Each of these problems has been previously encountered by developers, who, over time have provided solutions to these problems, each solution evolving and becoming more sophisticated as it is shared, eventually becoming a set of techniques which can be re-used in projects, allowing experience to pass through the developing community [3].

The identifying of an documentation of design patterns dates back to the 1980s [4], however, Bishop [5] states that design patterns were first introduced to the mainstream by the '*Gang of Four*' (GOF) also known as: Erich Gamma, Richard Helm, Ralph Johnson and John Vlissdes in their 1993 book '*Design Patterns: Elements of Reusable Object-Orientated Code*' [2]. The GOF specify and discuss 23 design patterns, of which, lay the foundations for design patterns as a concept, one which is considered to be an essential core of patterns to this day [5], [6]. Many more design patterns have been created and catalogued since the GOFs publication in 1995, however, the 23 are still the best known and popular [7].

In their historical work, the GOF [2] state that, in general, a design pattern has four essential elements:

- **Pattern Name** – A label which given to patterns to describe a design problem, its solutions and consequences.
- **A Problem** – The problem describes to the programmer when to use the pattern, explain the problem and its context.
- **A Solution** – Describing elements which form the design, their relationships, responsibilities and collaborations.
- **The Consequences** – The trade-offs and results of applying a pattern to a particular problem.

However, they argue that a developer's point of view can affect their interpretation of what is or isn't a pattern, meaning that, one person's design pattern is another's simple starting point. In addition to the four essential elements, the GOF also use a specific format to structure their patterns (table 1) which is based on the work by Peter Coad [8]. The GOF template is used to describe patterns using a consistent format by dividing the patterns into sections, thus making the patterns structure and information uniform making them easier to learn, compare and use [2]. It is worth noting that there are many design pattern formats and structures, including the *Alexandrian Form* [9] and the *Portland Form* [8], however, the GOF is by far the most influential [3].

<i>Section</i>	<i>Description</i>
Pattern Name & Classification	A patterns name must express the crux of the pattern concisely as it will become part of the vocabulary used in the design.
Intent	A statement of the patterns purpose. The point of the pattern should be visible at a glance.
Also Known As	Is the pattern known by any other names, if any?
Motivation	Illustration of the design problem through a scenario showing how classes and objects solve said problem helps the understanding of more abstract pattern descriptions.
Applicability	Where can the design pattern be applied? What problems does it address? How are the situations recognised?
Structure/Interaction	UML/Class/Interaction diagrams describing the relationships between the classes and objects in the pattern.
Participants	Objects and/or classes found in the design pattern and their responsibilities.
Collaborations	How participants carry out their responsibilities through collaboration.
Consequences	The trade-offs and results of applying a pattern to a particular problem.
Implementation	The details of the pattern solution. The downsides, hints or techniques to be aware of when implementing a pattern. Such as language specific problems.
Sample Code	Code samples which illustrate how the pattern may be implemented. This can be in any language, however, for this assignment, PHP.
Known Uses	Real life situations where the solution is/could be found. At least two examples should be included.
Related Patterns	What design patterns closely relate to the one being described. Some patterns will imply others or would have been inspired from others.

Table 1 - Describing Design Patterns GOF Format [2], [3]

2.2 Reasons for Using Design Patterns

Section 2.1 has discovered that design patterns define problems and describe solutions to these problems, which in turn, help developers solve common problems. However, there are many benefits to using design patterns.

Firstly, patterns help recognition of problems. Realising that there is a problem is often the first step towards finding a solution, secondly, once a problem is realised, design patterns provide solutions to problems using *proven* solutions, which can be copied easily with only minor changes required for implementation into a project [3], [7].

Design patterns also provide a vocabulary for describing how developers may approach problems; useful for communication between colleagues in meetings. By providing a commonly recognisable language to communicate in, developers' mobility and value of knowledge is increased [3], [5]. For example, if two developers (a & b), were planning the architectural design of a database logging system for a web project, developer **A** may suggest using the *factory* pattern, developer **B** disagrees and suggests the *observer* pattern. Developer B is correct; however, each developer knew exactly what the other was describing, therefore enhancing their interaction experience and speeding up the development process.

Zandstra [3] also adds that application of patterns is also useful because using one often leads to the use of another, adding that they demonstrate good use of object-orientated design. Additionally, design patterns are what forms the basis for PHP frameworks such as Laravel [10], Symfony [11] or CodeIgniter [12] along with other language frameworks, like C#'s Entity Framework. Maloriello [7] continues to add that design patterns allow for cohesive modules with limited coupling which in turn makes systems easier to maintain and understand.

However, Zandstra [3] states that design patterns have attracted some criticism through their overuse, particularly by inexperienced programmers, however, this is usually down to their use in situations or contexts where they are not required, meaning that patterns should be selected carefully for each problem as the details of implementation will vary greatly in respect to the context of the problem.

2.3 Examples of Patterns in Web Development

The following section of the report will look closely at two popular design patterns used in web applications: the *adapter* pattern and the *factory* pattern. These have been chosen down to popularity and common use in PHP applications. The ensuing sections will describe these patterns in detail, explain where they are commonly used whilst documenting their benefits and drawbacks. Screenshots of example code are included in this section of the report with a full code listing in the appendix and working code examples in an accompanying zip file.

2.3.1. Adapter Pattern

In Programming change is certain. Web applications or software are subject required to changes due to new requirements months after initial implementation. The application developed may have been designed for a particular client, data set or may just be dependent on an external API which is subject to change [13], [14].

When the problem of change arises, there are two solutions: either re-write all the code to accommodate the change or utilise the *adapter* pattern. The adapter pattern, also known as the *wrapper* pattern falls under the category of *structural* patterns [2], [5], [14], and can be used in two ways: with classes, using inheritance, or with objects using composition [15]. In programming, composition is preferred over inheritance [2], [16] due to its looser coupling and adherence to the single responsibility principle [17].

The intent of the adapter pattern is to translate an interface of one class into a compatible interface the client expects, allowing classes to work together that otherwise could not due to incompatibilities in interfaces by presenting the *wrapped* interface to clients whilst using the original interface [2], [6], [18]. The most common real-world visualisation of the adapter pattern is electrical adapters, to charge a mobile phone, for example (fig. 1). Most electrical sockets do not accept USB connections therefore an adapter is required to accept a USB charging cable to convert into a standard wall socket where: the USB cable is the client wanting to receive a service from the adaptee (socket), to achieve this a wrapper or common interface (adapter) is used to change the interface of the socket to one the USB expects. The main point being that both USB cable and the socket remain unchanged.

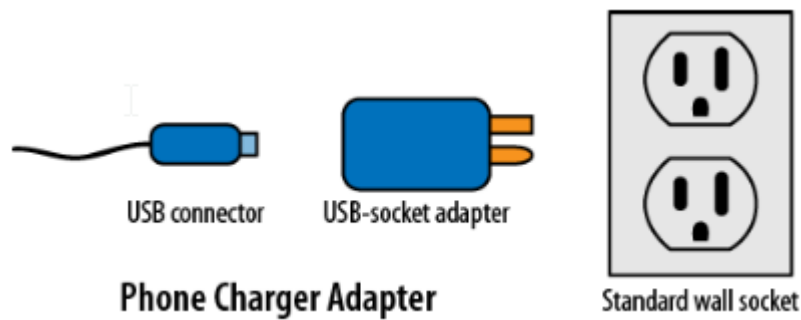


Figure 1 - Common Real-World Example of Adapters [15]

2.3.1.1 Adapter Pattern Example

The adapter pattern is one of the most commonly used patterns in web applications [19], due to the ever changing requirements of web applications. The following example demonstrates how the adapter pattern can be used to adapt web applications to accept more than one payment method without having to modify code throughout the source code. This example has been modified from Zala [14], who states that the use of this pattern was vital when PayPal changed their API method name to 'sendPayment' from 'payAmount', by utilising the adapter pattern just one wrapper interface needs to be created to avoid modifying each class the method is used.

The screenshot below shows the original *PayWithPayPal* class which uses a method called 'SendPayment', accepting a value from a simple form through variable *\$amount*.

```
class PayPal
{
    public function SendPayment($amount)
    {
        echo "Payment Amount of £" . $amount . " Sent via PayPal";
    }
}

$paypal = new PayPal();
$paypal->SendPayment($amount);
```

Payment Page

Payment Amount of £10 Sent via PayPal

However, if the API method name changed this would no longer work, requiring all references to this method to change throughout the application. The solution is the adapter pattern. Firstly, an interface is created containing a method named after the original in the *PayWithPayPal* class which will serve as the new payment function wrapper for the PayPal adapter, and any future payment methods.

```
interface IPaymentAdapter
{
    public function SendPayment($amount);
}
```

A new *PayWithPayPalNew* class is created with the new API *PayAmount* method, along with a *PayPalAdapter* class implementing *IPaymentAdapter* interface and its method *SendPayment*. A new *PayWithPayPalNew* object is the passed into the constructor. Now the same method names are used for both *PayWithPayPal* and *PayPalAdapter*.

```
class PayWithPayPalNew
{
    public function PayAmount($amount)
    {
        echo "Payment Amount of £" . $amount . " Sent via PayPal";
    }
}
```

```

class PayPalAdapter implements IPaymentAdapter
{
    // Attributes
    private $_payPalNew;

    // Constructor

    public function __construct(PayWithPayPalNew $_payPalNew)
    {
        $this->_payPalNew = $_payPalNew;
    }

    // Properties

    // Methods
    public function SendPayment($amount)
    {
        $this->_payPalNew->PayAmount($amount);
    }
}

$paypal = new PayPalAdapter(new PayWithPayPalNew());
$paypal->SendPayment($amount);

```

A new instance of the *PayPalAdapter* class is then initialised and the function *PayAmount* called from the *PayPalAdapter* class with the amount to be sent passed in from the form. In the index file, the *PayPalAdapter* class is then included upon submission of the form, instead of the *PayWithPayPal* class meaning that limited changes of code throughout the codebase are required.

```

<div class="row">
    <form action="#payForm" method="post" class="inputs">
        <input type="text" name="amount" placeholder="£0.00">
        <input type="submit" name="paySubmit" id="payForm" value="Pay">
    </form>
</div>

<br><br>
<div class="row">
    <div class="echoes">
        <?php

            if (isset($_POST["paySubmit"]))
            {
                $amount = $_POST["amount"];
            }

            if (isset($_POST["paySubmit"])) {
                include_once 'PaymentClasses/PayPalAdapter.php';
            }

```

Payment Page

Payment Amount of £1 Sent via PayPal

The method *SendPayment* present throughout the codebase can now be used, via the *IPaymentAdapter* interface to implement further payment type changes if required. Visa payments, for example. To allow this, a new *PayWithVisa* class needs to be created along with a *VisaAdapter* class following the same procedure as before. This time, the method is called *PayVisa*.

```
class PayWithVisa
{
    public function PayVisa($amount)
    {
        echo "Payment Amount of £" . $amount . " Sent via PayWithPayPal";
    }
}
```

With the *VisaAdapter* class following the same pattern as the *PayWithPayPalNew* cl

```

class VisaAdapter implements IPaymentAdapter
{
    // Attributes
    private $_visa;

    // Constructor
    public function __construct(PayWithVisa $_visa)
    {
        $this->_visa = $_visa;
    }

    // Properties

    // Methods
    public function SendPayment($amount)
    {
        $this->_visa->PayVisa($amount);
    }
}

$payWithVisa = new VisaAdapter(new PayWithVisa());
$payWithVisa->SendPayment($amount);

```

The index page can then be updated accordingly to allow users to select which payment they would like. Based on their selection a different adapter class is called whilst still utilising the original *SendPayment* method used when creating the application.

```

<form action="#payForm" method="post" class="inputs">

    <div class="form-check form-check-inline">
        <input type="radio" name="pType" id="paypal" value="paypal">
        <label class="form-check-label" for="paypal">PayPal</label>
    </div>
    <div class="form-check form-check-inline">
        <input type="radio" name="pType" id="visa" value="visa">
        <label class="form-check-label" for="visa">Visa</label>
    </div>
    <br><br>
    <input type="text" name="amount" placeholder="£0.00">
    <input type="submit" name="paySubmit" id="payForm" value="Pay">

</form>
</div>
<br><br>
<div class="row">
    <div class="echoes">
        <?php
            if (isset($_POST["paySubmit"]))
            {
                $amount = $_POST["amount"];

                if (isset($_POST["paySubmit"]) && $_POST["pType"] == "paypal") {
                    include_once 'PaymentClasses/PayPalAdapter.php';
                }
                else if (isset($_POST["paySubmit"]) && $_POST["pType"] == "visa") {
                    include_once 'PaymentClasses/VisaAdapter.php';
                }
            }
        </?php
    </div>

```

This can then be extended as required to accept new payment types as and when required by simply creating two new classes for each payment type. This pattern can be used with a variety of services.

Payment Page

☐ PayPal ☒ Visa

9

Payment Amount of £9 sent via Visa

The full code accompanying this report extends this example further.

2.3.2 The Template Method Pattern

The template pattern is part of the *behavioural* group of patterns [20], which use composition over inheritance. Other behavioural patterns also include: The *Interpreter*, *Mediator*, *Chain of Responsibility*, *Observer*, *Strategy*, *Command*, *State*, *Visitor*, and *Iterator*. These patterns describe the communication between objects and classes as well as the patterns. Their concern is with the algorithms and assignment of an object's responsibilities [2].

The template pattern, or *template method* pattern as it is also known, is a common pattern used throughout web applications, and is simple to implement through use of an abstract class and a concrete class. Many use the analogy of the '*Hollywood Principle*' to describe its intent: "*don't call use, we'll call you*" as, by using abstraction it is called by a template parent classes whilst letting the subclasses finish or redefine the algorithms behaviour, without changing its structure [2], [20].

In practice, the motivations for using the template pattern are to avoid code repetition by grouping together common behaviour into a common class, or as Opdyke and Johnson [21] state "*refactoring to generalize*" by separating identified differences into new operations. Bishop [5] adds that the template pattern is useful with the *strategy* pattern and any system where deference of basic operations to other classes will be beneficial.

2.3.2.1 Template Pattern Example

A real-world example would be that of posting to different social networks, like Facebook or Twitter, where both platforms would require users to log in with their username and password and post a message. This would require the use of two classes, each containing very similar attributes and methods. The screenshot below demonstrates a class which could be used for posting status updates, however, the methods contained in this class could be easily used in other classes to post updates on twitter or other social network

```

class Facebook
{
    // Attributes
    protected $_username;
    protected $_password;

    // Constructor
    public function __construct($username, $password)
    {
        $this->_username = $username;
        $this->_password = $password;
    }

    // Properties

    // Methods
    public function LogIn($userName, $password)
    {
        // Log-in code
    }

    public function SendMessage($message)
    {
        // Code to send message
    }

    public function logOut()
    {
        // Log-out code
    }

    public function PostMessage($message)
    {
        // Code to post message.
    }
}

```

This is where the template pattern is useful: there are methods which would be present in both the *Facebook* class and the *Twitter* class, therefore it is possible to use an abstract parent class which defines the algorithms skeleton whilst delegating some steps of implementation to the subclasses (Facebook/Twitter) without changing the initial structure of the algorithm which posts a message to each social network.

First, a parent *template* abstract class must be created to house all the boilerplate code which will be used by the subclasses.

The abstract class *SocialNetwork* contains the *Post* function, which will be used by all social networks, however, it also contains all the template methods: *LogIn*, *SendMessage* and *LogOut* which are declared abstract for the subclasses to override them with their individual implementations, such as their different API calls.

```
abstract class SocialNetwork
{
    // Attributes
    protected $userName;
    protected $password;

    // Constructor
    public function __construct($userName, $password)
    {
        $this->userName = $userName;
        $this->password = $password;
    }

    // Methods
    public function Post($message)
    {
        if ($this->LogIn($this->userName, $this->password)) {
            $result = $this->SendMessage($message);
            return $result;
        }

        return false;
    }

    // Abstract Methods
    abstract public function LogIn($userName, $password);

    abstract public function SendMessage($message);
}
```

Then, the subclasses *Facebook* and *Twitter* can just contain their individual *LogIn* and *SendMessage* methods whilst extending the *SocialMedia* class containing the commonly shared methods. As the functions *LogIn* and *SendMessage* are declared abstract in the *SocialMedia* class both the *Facebook* and *Twitter* classes' *LogIn* and *SendMessage* methods can override them with their own implementation.

```

class Facebook extends SocialNetwork
{
    // Attributes

    // Constructor

    // Properties

    // Methods
    public function LogIn($userName, $password)
    {
        echo "Facebook: '". $this->userName.'" has logged in successfully.";
        echo "<br>";
        return true;
    }

    public function SendMessage($message)
    {
        echo "<br>";
        echo "<br>";
        echo "Facebook: '". $this->userName.'" has posted '". $message.'" \n";
        return true;
    }
}

```

A form is then created to allow input from the user and stored in variables using the `$_POST` super-global. A new *Facebook* or *Twitter* object is then created, passing in the username and password from the form, allowing log-in. This is stored in the variable *\$facebookPost* or *\$twitterPost*. An object operator is then used to access the *Post* method in the *SocialMedia* class which posts the message entered into the text area.

```

<?php
    if (isset($_POST["logInPost"]))
    {
        $password = $_POST["password"];
        $userName = $_POST["username"];
        $message = $_POST["message"];
        $facebookPost = new Facebook($userName, $password);
        $facebookPost->Post($message);
    }
    else
    {
        die ("Post not sent");
    }
?>

```



Enter a post...

Post

Facebook: 'fbuser' has logged in successfully.

Facebook: 'fbuser' has posted 'This is a successful test of the Template Method pattern.'



Tweet...

Post

Twitter: 'twuser' has logged in successfully.

twuser' has Tweeted: 'This is a successful test of the Template Pattern using the Twitter class.'

Full code for the following example will be included in the final report showing the two separate classes and how they override the abstract class. The example shown is modified from an example found at *Refactoring.Guru* [22]. For the purpose of the example the username and password have been hardcoded using HTML. Upon completion of this example, further code recognition was discovered; refactoring of the HTML pages to allow a single page to post to either Facebook or Twitter is possible via a radio button or like, to avoid repetition.

Web Application

3.1 Description of Web Application

The web application produced is designed for Swansea City football fans to read about news at the club. The website also allows fans to sign up for an account to view extra features, such as, viewing each team member through a search engine and past and upcoming fixtures through a pageable interface. The application also allows administrators to log-in and perform CRUD queries, via SQL on PHP MyAdmin database for all tables.

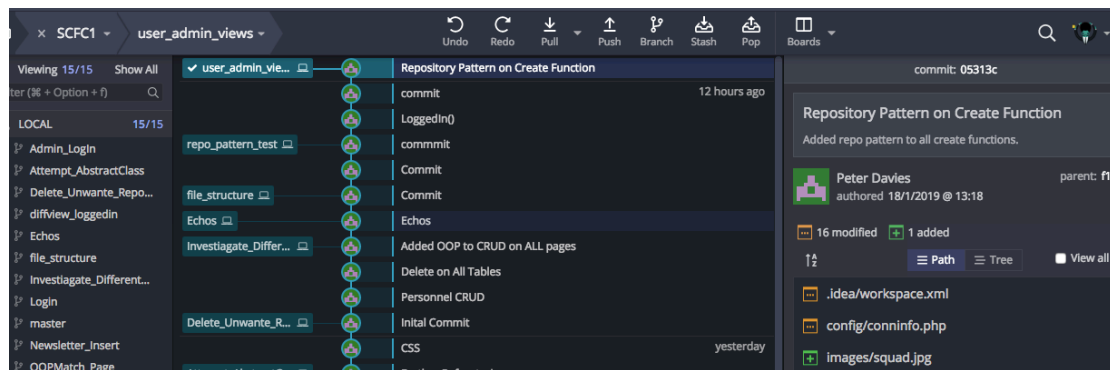
The HTML, CSS and some queries have been re-used as a template from a 2017 assignment produced by Davies [23] for a previous PHP web development assignment. Although the applications genre, design and some of the database structure remain the same, the content, images, programming techniques and file structure are vastly different as well as the addition of new database columns and a table. The web application submitted has been re-developed, with an attempt at an object-orientated approach with the use of design patterns whilst also using Git version control, as opposed to unorganised procedural programming. The original application can be supplied for comparison, should any plagiarism issues arise.

3.2. Git Version Control

Git version control is a system repository which documents changes made to files or sets of files over time so that different versions can be recalled at any point in time. Git repositories are generally used for development project, however, it can be used for any type of file [24].

Git aids in development of applications as it allows developers to attempt new features which may involve modifying vast amount of code throughout a project without the fear of forgetting what has be modified. If an error is made, or code does not work, the project can revert back to the previous working version, allowing developers free reign to try out different techniques and features.

Git was used for this project and a Git GUI was used in the form of GitKraken [25]. A version history can be views through any Git GUI or CLI (command line interface).

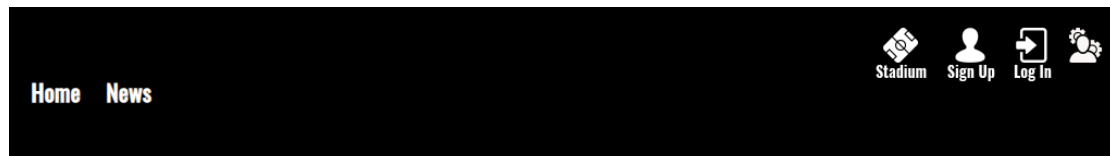


3.3 Features of Web Application

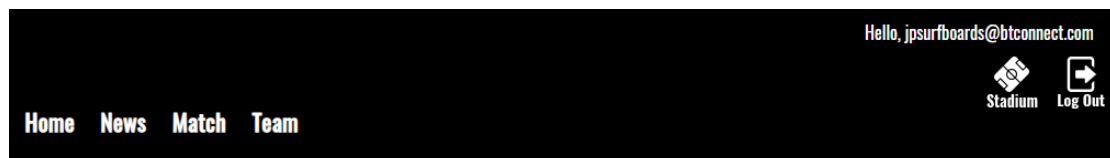
The following section of the report outlines some of the features the web application offers.

3.3.1 Restricted Views

The web application offers a limited view to users who are not members or admins. Upon first visit to the website the users view is restricted, showing only the following links in the header:



If a user is registered as a regular member, they are authorized to access the hidden features of the application. The header displays these options upon signing in:



Features 'Match' linking to a pageable fixture display and feature 'Team' linking to a search engine of Swansea City Players is now available. This is achieved through the *session* variable which is set to the users email upon password verificatio

```

if (password_verify($password, $result["password"]))
{
    $_SESSION["userLoggedIn"] = $result["eMail"];
    return true;
}

```

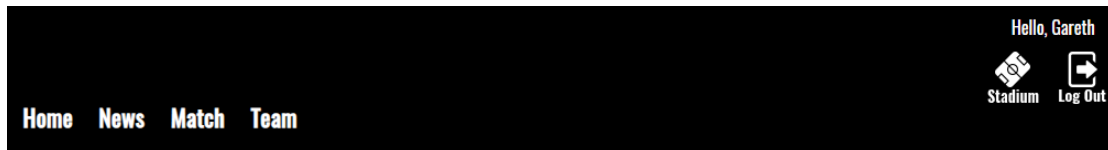
The session variable is then used in a function in the *user* class, which checks if a user is logged in. It is then called in the header include file and used as a condition to display varying HTML.

```

public function LoggedIn()
{
    if (isset($_SESSION["userLoggedIn"]) && $_SESSION["userLoggedIn"] != "")
    {
        return true;
    }
}

```

The same session variable is used to display the current user logged in. The same technique is used to display different views to the administrators and display the administrators name:



It is understood from the assignment brief that the display of different views to users with different authorisation levels can be achieved using design patterns, namely the *decorator*, *composite* or *strategy* patterns, however, due to time constraints on this project the research required to implement this design pattern could not be carried out. These patterns will require further research for future projects.

3.3.2 Member Registration & Login

To allow members to login to the application, registration is first required. This is achieved, again through the *user* class.

```
public function Register($dbh, $fName, $sName, $eMail, $password)
{
    try {
        $hashPassword = password_hash($password, algo: PASSWORD_DEFAULT);

        $insert = $dbh->prepare("INSERT INTO members(FirstName, Surname, eMail, password)
                                VALUES (:FirstName, :Surname, :eMail, :password)");

        $insert->bindParam(':FirstName', $fName);
        $insert->bindParam(':Surname', $sName);
        $insert->bindParam(':eMail', $eMail);
        $insert->bindParam(':password', $hashPassword);
        $insert->execute();

        ?>
        <div class="news-success-msg">
            <p>
                <? echo "You have been successfully registered!";
                header( string: 'Refresh:1; url:../SCFCL/login.php');
                ?>
            </p>
        </div>
        <?
    }
    catch (PDOException $e) {
        throw $e;
    }

    $this->Notify($eMail);
}
```

A web form is used to allow users to enter their data, the fields are then checked through a series of validation methods. Upon verification, the posted using the `$_POST` variable. A *try/catch* block is then used firstly, to call the function *RegisteredCheck()* which queries the database for existing records which match the entered email address, if no records exist, the function *Register()* is called with the post data passed in to the PDO insert query which also binds the parameter to prevent SQL injection. The function contains the added security measure of hashed passwords using a built in PHP function. If the query is unsuccessful, an error message is displayed through the PDO *getMessage()* error function. The user is then redirected to login.

The user class in this application contains many functions relating to the user, however, this could be in violation of the single responsibility principle. It is understood that splitting this class into further classes would be beneficial, such as, separating the log-in and registering functions into a separate interface which each separate user (member, admin) could implement.

3.3.3 Player Searches

Members can search players using keywords and filters.



SEARCH SWANSEA CITY PLAYERS USING THE SEARCH BOX BELOW...

ENTER SEARCH

SEARCH BY...

☐ Surname ☐ First Name ☐ Player Position ☐ Player Number

SEARCH

A function *Search()* is given in the *LogSearch* class and called to query the database with the users required parameters.

```
public function Search($dbh, $filter, $keyword)
{
    try {
        $searchPlayers = $dbh->prepare("SELECT * FROM clubpersonnel, players
        WHERE clubpersonnel.EmployeeNO = players.EmployeeNO AND $filter like '$keyword'");
        $searchPlayers->execute();
        return $searchPlayers->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        echo $e->getMessage();
    }
}
```

The results from this query are then returned in an array of results using the PHP statement *fetchAll()* and indexed by column name using by using the *FETCH_ASSOC* parameter. A *foreach* loop is then used to iterate through the array results with *echo* used to display the relevant fields

```

foreach($data as $row)
{
    <div class="col-12 php-res-player">
    <div class="php-res-name">
    <p><?
        echo $row["FirstName"]." ";
        echo $row["Surname"]." ";
    ?>
    </p>
    </div>
    <div class="php-res-img">
    <? $pic=$row["Photo"];
    if($pic=="") //If picture isn't there
    {><?>
    else // Otherwise show the image in the database
    {>" width="200" height="250"><?>
    </div>
}

```

3.3.4 Pageable Fixtures

Fixtures and results are displayed using a paging script provided by Thomas [26] for the 2017 assignment.

FIXTURES & RESULTS

SWANSEA SWANS  VS  13-12-2017 15:00 PM 0 - 4	SWANSEA EVERTON  VS  18-12-2017 20:00 PM 3 - 1	SWANSEA C. PALACE  VS  23-12-2017 15:00 PM 1 - 1
--	--	--

The implantation of this requires two functions; *GetFixtures()* to return an array of results through PHP's *rowCount()* function which returns rows which are affected by the SQL statement executed by the PDO object.

```

public function GetFixtures($dbh)
{
    try {
        $getFixtures = $dbh->prepare("SELECT * FROM fixtures, clubs WHERE fixtures.ClubID = clubs.ClubID");
        $getFixtures->execute();
        $numRows = $getFixtures->rowCount();
        return $numRows;
    } catch (PDOException $e) {
        echo $e->getMessage();
    }
}

```

The returned array is then conditionally compared to more than 0 results. If the results are more than 0, the *PageFixtures()* method is called which performs another SQL query but with limiting parameters .

```

public function PageFixtures($dbh)
{
    $limit = 3;

    if (empty($page)) {
        $page = 0;
    }

    try {
        $pagedFixtures = $dbh->prepare("SELECT * FROM fixtures, clubs
        WHERE fixtures.ClubID = clubs.ClubID LIMIT $page, $limit");
        $pagedFixtures->execute();

        $count = 1 + $page;

        return $pagedFixtures;
    } catch (PDOException $e) {
        $e->getMessage();
    }
}

```

The results from this query are then echoed out to display data from the fixtures table.

3.4 Design Patterns Used

The following section of the report discusses the design patterns used and attempted in the application.

3.4.1 Observer Design Pattern

The GOF [2] state the Observer design pattern “*defines and maintains a dependency between objects*” meaning that when one object changes its state the other objects are notified [5]. In web application terms, this could be to notify a webmaster about failed log-in attempts or new registered users, or to write logs to a database. In the web application submitted, the observer pattern observes two objects for change of state: the register object and the search object.

The register object is observed for new users signing up to the application. To do this an interface *IObserver* is created with two methods *NewUserReg()* and *SearchLog()* the latter used for the second observer.

```
interface IObserver
{
    public function NewUserReg($newUser, $data);

    public function SearchLog($dbh, $userEmail, $keyword, $filter, $dateTime);
}
```

A second interface, *INewUserObserver* is then created to attach, detach and notify. The user class then implements the *INewUserObserver* class and its functions. The *LogNotifier* object is then attached is attached to the user class in the *conninfo* configuration file.

```
// Attach observers
$user->Attach(new LogNotifier());
$logSearch->Attach(new LogNotifier());
```

The *Attach()* function in the user class then stores an array of observed state changes. The *Notify()* function then iterates the results and notifies the observers with the new user information, in this case *NewUserReg()* found in the *IObserver* interface.

```

private $_newUser = array();

public function Attach($newUser)
{
    $this->_newUser[] = $newUser;
}

public function Detach($newUser)
{
    // TODO: Implement Detach() method.
}

public function Notify($eMail)
{
    foreach ($this->_newUser as $nUser)
    {
        $nUser->NewUserReg($this, $eMail);
    }
}
}

```

A class *LogNotifier()* is then created to implement the *IObserver* interface and perform the operations.

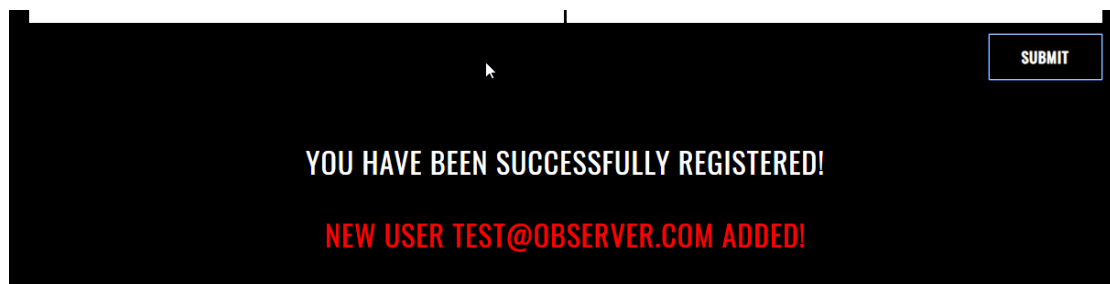
```

public function NewUserReg($newUser, $data)
{
    // TODO: Get mail server working through WAMP
    //$mail = mail("pdavies.079868@student.uwtsd.ac.uk", "New SCEC Archive Member!", "A new
    ?><div class="error-message"><p><? echo 'New User ' . $data . 'Added!'; ?></p></div><?
}

```

3.4.1.1 Testing New User Observer

In this example, an email should be sent to the webmaster notifying a new registration, however, due to development on a local server, connecting to an STMP mail server requires lengthy configuration. To demonstrate this functionality, the data observed is echoed inste



The second observer implemented observes member searches, logging their username, the keyword searched, the filter applied and the date and time to a database table. For this, a new observer interface is created called *ISearchObserver* which is implemented by the *LogSearch* class which handles all the search related functions.

```
// Observer code
private $_newSearchLog = array();

public function Attach($newSearchLog)
{
    $this->_newSearchLog = $newSearchLog;
}

public function Detach($newSearchLog)
{
    // TODO: Implement Detach() method.
}

public function Notify($userEmail, $keyword, $filter, $dateTime)
{
    foreach ($this->_newSearchLog as $sLog)
    {
        $sLog->SearchLog($this, $userEmail, $keyword, $filter, $dateTime);
    }
}
```

Again, the log notifier object is attached to the *LogSearch* class, storing each new search in an array which is then iterated by the *Notify()* function, calling the *SearchLog()* function which inserts a record of the member search into the database.

```

public function SearchLog($dbh, $userEmail, $keyword, $filter, $dateTime)
{
    try {
        $searchLog = $dbh->prepare("INSERT INTO searchlogs (userEmail, keyword, filter, dateTime)
                                   VALUES (:userEmail, :keyword, :filter, :dateTime)");
        $searchLog->bindParam(':userEmail', $userEmail);
        $searchLog->bindParam(':keyword', $keyword);
        $searchLog->bindParam(':filter', $filter);
        $searchLog->bindParam(':dateTime', $dateTime);
        $searchLog->execute();
    } catch (PDOException $e) {
        echo $e->getMessage();
    }
}

```

3.4.1.2 Testing Search Log Observer

SEARCH BY...
☐ Surname
☐ First Name
☒ Player Position
☐ Player Number

LUKASZ FABIANSKI



POSITION: GOALKEEPER
NUMBER: 1
AGE: 33

KRISTOFFER NORDFELDT



POSITION: GOALKEEPER
NUMBER: 13
AGE: 29

ERWIN MULDER



POSITION: GOALKEEPER
NUMBER: 25
AGE: 29

☐ Edit
☐ Copy
☐ Delete
10 jpsurfboards@btconnect.com Goalkeeper PlayingPosition 2019-01-17 22:08:16

3.4.2 Repository Pattern

The repository pattern's use has been attempted in this application, however, two examples of its implementation are provided as it was first misunderstood. Although examples are provided, further understanding of this pattern is required, as it is not completely understood.

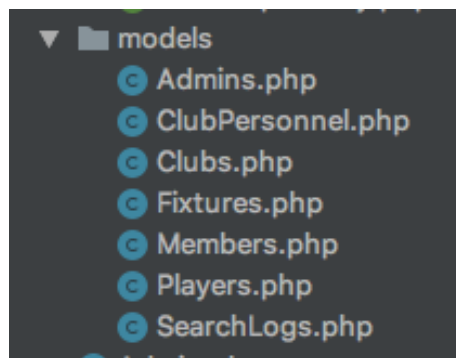
The repository pattern provides an abstraction layer to allow an application to function using simple abstraction from an interface resembling a collection. CRUD queries are then performed using simple methods in the interface, removing the need to handle database concerns, aiding to achieve loose coupling and keeping objects ignorant of one another [27]. A repository is used to allow an application to populate memory with domain entities from the database, once in memory, these entities can be changed and persisted back to the database through transactions [28] (SQL queries in this application).

Microsoft [29] state that the repository pattern should be used to achieve one or more of the following objectives:

1. To maximise the amount of code which can be automatically tested and to isolate the data layer.
2. To access a data source from as many locations and to apply a centrally managed, consistent access, rules and logic.
3. To implement and centralise a caching strategy for the data source.
4. To improve codes maintainability and readability by separating business logic from data or service access logic.
5. To use business entities which are strongly typed to identify problems at compile time over run time.
6. To associate a behaviour with the related data, e.g. calculate fields or enforce relationships or business rules between elements of data within an entity.
7. To supply a domain model to simplify complex business logic.

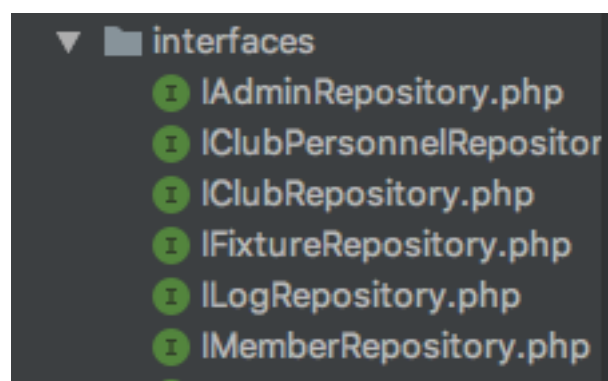
It is understood that the uses of the repository pattern in the presented application match with points 4 and 7. Point four referring to the separation and organisation of code throughout the application, into different folders, classes and models. Point 7 referring to the use of models to represent the entities in the database to simplify the business logic.

The first example of the repository pattern implemented consisted of a model for each of the entities in the database. Classes were then created which contained all the transaction logic for these models. It is worth noting that naming of the classes and models has been confused, in the submitted web application: *Admins* model should be *Admin* and vice versa. This applies to all the classes and models represented in the project.



```
class Admins
{
    public $adminId;
    public $eNo;
    public $password;
    public $pHint;
}
```

Interfaces were then created for the corresponding classes to implement their functions.



Each interface containing a set of methods to CRUD the data from the database, each method has the corresponding model passed in to allow data to be queried from the database using the model.

```

interface IAdminRepository
{
    public function Create($dbh, Admins $admins);

    public function Read($dbh, Admins $admins);

    public function Update($dbh, Admins $admins);

    public function Delete($dbh, Admins $admins);
}

```

The model object *Admins* is then passed into the *Admin* object to allow the *Admin* object use of the variables representing columns in the database. The interface was then implemented by the *Admin* class and SQL queries retrieve data from the database.

```

$admins = new Admin(new Admins());

```

```

public function Read($dbh, Admins $admins)
{
    try {
        $getAdmin = $dbh->prepare("SELECT * FROM admin");
        $getAdmin->execute();
        $adminArray = $getAdmin->fetchAll(PDO::FETCH_ASSOC);
        return $adminArray;
    } catch (PDOException $e)
    {
        $e->getMessage();
    }
}

```

Each of the entity's classes have their own interface, with their corresponding model passed in and implemented. At first, it was believed that repeating this much code was a bad thing, however, further research led to the belief that although code repetition was rife, the reward is that the logic which queries the database is now separated from the application, meaning that with very little change, a different database could now be used implementing the same methods from each of the interfaces. The code for this first example can be found in the Git repository *Admin_Login* branch.

The second example of the repository pattern resulted from the belief that too much code was being repeated. In this example, one interface called *IRepositoryInterface* was created of

which all the classes implemented, thus reducing code repetition, however, passing the model into the function proved difficult as PHP expects the same method declaration in an interface's child class as in the interface. As each model to be passed in was different, this proved difficult to understand.

```
interface IRepositoryInterface
{
    public function Create($dbh, $data);

    public function Read($dbh);

    public function Update($dbh);

    public function Delete($dbh);
}
```

The solution was to pass in a variable *\$data* as PHP does not specify what type of data is stored in variables at compile time, but is decided by PHP at runtime depending on the context of which the variable is used [30]. This means that when the form is submitted, the variables used in the query can be set to the corresponding column names in the model by using the *\$_POST* variable.

```
$admins->eNum = $_POST["EmployeeNo"];
$admins->pWord = $_POST["Password"];
$admins->pHint = $_POST["PasswordHint"];
```

The variable *\$admins* representing an instance of the model *Admins*. This post data is then passed back to the function through the *\$data* object variable and assigned to each variable used in the SQL query.

```

$eNum = $data->eNum;
$password = $data->pWord;
$pHint = $data->pHint;

$insert = $dbh->prepare("INSERT INTO admin(Employe
$insert->bindParam(':EmployeeNo', $eNum);
$insert->bindParam(':Password', $password);
$insert->bindParam('PasswordHint', $pHint);
$insert->execute();
return true;

```

This function *Create()*, containing the SQL is then called in the view page to display the queries results.

```

$admins->Create($dbh, $admins);

```

It is worth noting that in the submitted web application, the use of this technique to query data has only been used on the *Create()* functions due to time constraints.

3.4.3 The Façade Pattern

The façade pattern is a method of providing one simple interface to complex systems by providing a high-level view of the sub-systems whilst hiding the details [3], [5]. A simple real-world example provided by Bishop [5] is that of an Amazon purchase: known users can purchase items with one click as opposed to unknown users who have to trudge through the full checkout system. This means that the one-click purchase system shown to known users is forming a façade over the full system that lays beneath.

The web application submitted does not use the façade pattern due to time constraints, however, a very simple example is demonstrated here and provided in the accompanying zip files. The example is modified from example code by Benharosh [31].

Often on webpages a button to share information or messages is found which simultaneously share a message to a number of social media platforms, this is done through using the façade

pattern; a high-level view is concealing the sub systems of each individual social media platform beneath it.

First a façade class is created which will reference each class containing each individual logic for posting messages to each platform. The objects are then injected via the constructor, and one function combines the three functions contained in the social media classes.

```
class shareFacade {  
  
    private $_twitter;  
    private $_facebook;  
    private $_pintrest;  
  
    function __construct($twitterObj, $facebookObj, $pintrestObj)  
    {  
        $this->_twitter = $twitterObj;  
        $this->_facebook = $facebookObj;  
        $this->_pintrest = $pintrestObj;  
    }  
  
    function Share($tweet, $post, $share)  
    {  
        $this->_twitter->Tweet($tweet);  
        $this->_facebook->Post($post);  
        $this->_pintrest->Share($share);  
    }  
}
```

Each social media classes logic contains a simple echo which displays a message entered into a text box through a HTML form.

```
class Pintrest {  
  
    public function Share($share)  
    {  
        echo "Your message: - " . $share. " - has been shared on Pintrest!";  
    }  
}
```

The objects are then created from the social media classes and passed into another variable *\$shareObj* which is set to store the creation of the new façade object with the social media objects passed in. The *Share()* method from the façade is then called on the *shareObj* variable with each posted message passed in.

```
if (isset($_POST["sPost"]))
{
    $submitPost = $_POST["sPost"];
    $tweet = $_POST["text"];
    $post = $_POST["text"];
    $share = $_POST["text"];
    $twitterObj = new Twitter();
    $facebookObj = new Facebook();
    $pintrestObj = new Pintrest();

    $shareObj = new shareFacade($twitterObj, $facebookObj, $pintrestObj);

    $shareObj->Share($tweet, $post, $share);
}
```

This means that rather than calling three separate methods, the façade allows just call of just one method *Share()* to post to all three social media services, with the user being none the wiser of the unlaying sub-system underneath

Facade Pattern

Enter a post...

Post

Your Tweet - TEST POST - has been Tweeted!

Your post: - TEST POST - has been posted to Facebook!

Your message: - TEST POST - has been shared on Pintrest!

4. Evaluation

The web application submitted utilises some basic object orientated programming techniques through the use of some design patterns. However, should the assignment be repeated, with more time, a different approach would be taken.

Firstly, re-using code from a previous project probably took more time than expected and starting from scratch, focusing on just the back-end code would have allowed more time for research into and implementation of better or more design patterns. That said, the web application does look good! Although, not mobile first, view on a big screen for best results!

Additionally, starting from scratch would have allowed a better organised project, utilising a better file structure. At present, the files are unorganised, badly named and in the wrong places. Each view, for example, should be in the 'views' folder, however, having decided to create this folder at the end of the project, renaming every file which referenced a view was too difficult to implement.

More time and planning would also have led to more appropriately named class files, at present, they are all renamed confusingly. Time was also a big factor when considering the database for this project, as previously mentioned, it is a modified one from a previous assignment. As useful as this has proved in saving time, a new one would have been preferred.

On the whole the application has given a good opportunity to practice OOP programming, and learn about design patterns with much has being learnt from the first assignment for advanced web development.

References

- [1] C. Alexander, S. Ishikawa, M. Silverstain, M. Jacobson, I. Fiksdahl-King, and S. Angel, "A Pattern Language." Oxford University Press, 1977.
- [2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Indianapolis: Addison-Wesley, 1995.
- [3] M. Zandstra, *PHP Objects, Patterns, and Practice*, 4th ed. Berkeley : Apress, 2008.
- [4] J.-M. Jézéquel, M. Train, and C. Mingins, *Design patterns and contracts*. Addison-Wesley, 2000.
- [5] J. Bishop, *C# 3.0 Design Patterns*. O'Reilly, 2007.
- [6] E. Freeman and E. Freeman, *Head First Design Patterns -O'Reilly (2008)*. O'Reilly, 2008.
- [7] J. Maloriello, "What Are Design Patterns and Do I Need Them? - Developer.com," *Developer*, 2002. [Online]. Available: <https://www.developer.com/design/article.php/1474561/What-Are-Design-Patterns-and-Do-I-Need-Them.htm>. [Accessed: 10-Jan-2019].
- [8] W. Cunningham, "About the Portland Form," *Portland Form*. [Online]. Available: <https://c2.com/ppr/about/portland.html>. [Accessed: 10-Jan-2019].
- [9] "Alexandrian Form." [Online]. Available: <http://wiki.c2.com/?AlexandrianForm>. [Accessed: 10-Jan-2019].
- [10] "Laravel - The PHP Framework For Web Artisans." [Online]. Available: <https://laravel.com/>. [Accessed: 27-Apr-2018].
- [11] "Symfony, High Performance PHP Framework for Web Development." [Online]. Available: <https://symfony.com/>. [Accessed: 10-Jan-2019].
- [12] "CodeIgniter Web Framework." [Online]. Available: <https://codeigniter.com/>. [Accessed: 10-Jan-2019].
- [13] S. Jakowicz, "Adapter Pattern in PHP | Simon Jakowicz," 2014. [Online]. Available: <https://www.jakowicz.com/adapter-pattern-in-php/>. [Accessed: 10-Jan-2019].
- [14] A. Zala, "Design Patterns: The Adapter Pattern," *Code*, 2014. [Online]. Available: <https://code.tutsplus.com/tutorials/design-patterns-the-adapter-pattern--cms-22262>. [Accessed: 10-Jan-2019].
- [15] W. Sanders, *Learning PHP Design Patterns*, First. 2013.
- [16] K. Knoernschild, *Java design : objects, UML, and process*. Addison-Wesley, 2002.
- [17] R. C. Martin, "The Single Responsibility Principle," *Clean Code*, 2014. [Online]. Available: <https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleResponsibilityPrinciple.html>. [Accessed: 10-Jan-2019].
- [18] S. Alletti, "Design patterns: Adapter in PHP." [Online]. Available: <https://stefanoalletti.wordpress.com/2017/03/20/design-patterns-adapter-in-php/>. [Accessed: 10-Jan-2019].
- [19] I. Vorobiov, "The 5 Most Common Design Patterns in PHP Applications," *Medium*, 2018. [Online]. Available: <https://medium.com/@ivorobioff/the-5-most-common-design-patterns-in-php-applications-7f33b6b7d8d6>. [Accessed: 10-Jan-2019].
- [20] D. Liebler, *Design Patterns PHP Documentation*, Release 1. 2018.
- [21] R. E. Johnson and W. F. Opdyke, *Object technologies for advanced software : first JSSST international symposium, Kanazawa, Japan, November 4-6, 1993 : proceedings*. Springer-Verlag, 1993.
- [22] "Template Method in PHP," *Refactoring Guru*. [Online]. Available: <https://refactoring.guru/design-patterns/template-method/php/example>. [Accessed: 13-Jan-2019].
- [23] P. Davies, "Website with Dynamic Content using Open-Source Technologies," Swansea, 2017.

- [24] "Git - About Version Control," *Git SCM*. [Online]. Available: <https://git-scm.com/book/en/v1/Getting-Started-About-Version-Control>. [Accessed: 18-Jan-2019].
- [25] "Git Client - Glo Boards | GitKraken." [Online]. Available: <https://www.gitkraken.com/>. [Accessed: 18-Jan-2019].
- [26] "Gaynor Thomas - University of Wales Trinity Saint David." [Online]. Available: <http://www.uwtsd.ac.uk/staff/gaynor-thomas/>. [Accessed: 10-Dec-2017].
- [27] "Repository Pattern | DevIQ," *Dev IQ*. [Online]. Available: <https://deviq.com/repository-pattern/>. [Accessed: 18-Jan-2019].
- [28] "Designing the infrastructure persistence layer | Microsoft Docs," *Microsoft Developer Network*, 2018. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design>. [Accessed: 18-Jan-2019].
- [29] "The Repository Pattern | Microsoft Docs," *Microsoft Developer Network*, 2017. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649690\(v=pandp.10\)](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649690(v=pandp.10)). [Accessed: 18-Jan-2019].
- [30] "PHP: General Information - Manual," *php.net*. [Online]. Available: <http://php.net/manual/en/faq.general.php>. [Accessed: 14-Jan-2018].
- [31] J. Benharosh, "The Facade Pattern," *PHP Enthusiast*. [Online]. Available: <https://phpenthusiast.com/blog/simplify-your-php-code-with-facade-class>. [Accessed: 18-Jan-2019].

Bibliography

- Pokemon Go: Is the hugely popular game a global safety risk? - BBC News. (2016). Retrieved January 18, 2019, from <https://www.bbc.co.uk/news/world-36854074>
- Think you understand the Single Responsibility Principle? (n.d.). Retrieved January 17, 2019, from <https://hackernoon.com/you-dont-understand-the-single-responsibility-principle-abfdd005b137>
- database - Proper Repository Pattern Design in PHP? - Stack Overflow. (n.d.). Retrieved January 15, 2019, from <https://stackoverflow.com/questions/16176990/proper-repository-pattern-design-in-php>
- Tschabitscher, H. (2018). Configuring PHP to Use Remote SMTP Server to Send Mail. Retrieved January 15, 2019, from <https://www.lifewire.com/how-to-configure-php-remote-smtp-server-for-sending-mail-1171190>
- Johnson, R. E., & Opdyke, W. F. (1993). *Object technologies for advanced software : first JSSST international symposium, Kanazawa, Japan, November 4-6, 1993 : proceedings. Proceedings of the First JSSST International Symposium on Object Technologies for Advanced Software*. Springer-Verlag. Retrieved from <https://dl.acm.org/citation.cfm?id=709889>
- Ahmed, S. (2015). Adapter Design Pattern :: Code In PHP. Retrieved January 11, 2019, from <https://codeinphp.github.io/post/adapter-design-pattern/>
- Sweat, J. E. (2004). *PHP / Architect's Guide to PHP Design Patterns*. Retrieved from <http://www.phparch.com>
- Adapter in PHP. (n.d.). Retrieved January 10, 2019, from <https://refactoring.guru/design-patterns/adapter/php/1100#top>
- Quenneville, J. (2016). Composition vs Inheritance. Retrieved January 10, 2019, from <https://robots.thoughtbot.com/reusable-oo-composition-vs-inheritance>
- Adapter Pattern - GeeksforGeeks. (n.d.). Retrieved January 10, 2019, from <https://www.geeksforgeeks.org/adapter-pattern/>
- Benharosh, J. (2015). The adapter pattern in PHP | PHPenthusiast. Retrieved January 10, 2019, from <https://phpenthusiast.com/blog/the-adpater-design-pattern-in-php>
- Jézéquel, J.-M., Train, M., & Mingins, C. (2000). *Design patterns and contracts*. Addison-Wesley. Retrieved from <https://dl.acm.org/citation.cfm?id=555100>
- Fowler, M. (2003). *Patterns of Enterprise Application Architecture. The Addison-Wesley signature series*. Addison-Wesley. <https://doi.org/10.1119/1.1969597>

Herrington, J. D. (2006). *Five common PHP design patterns*. Retrieved from <https://www.ibm.com/developerworks/opensource/library/os-php-designptrns/os-php-designptrns-pdf.pdf>

Appendix A

Due to the large code base for this project, the code will be posted in hierarchical order, starting with the first folder in the project *config*. The CSS and other unimportant files will not be included.

Config > Conninfo.php

```
<?
// NEW CONNECTION OBJECT
$dsn="mysql:host=localhost;dbname=scfc1"; //server & database name
$dbUserName="root"; //server username
$dbPassword="root"; //server password ** WAMP = "p079868" / MAMP = "root"

try {
    $dbh = new PDO($dsn, $dbUserName, $dbPassword);
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo 'Connection Failed: ' . $e->getMessage();
}

// ** Includes ** //
require_once '../SCFC1/src/User.php';
require_once '../SCFC1/src/AdminUser.php';
require_once '../SCFC1/src/LogSearch.php';
require_once '../SCFC1/src/Fixture.php';
require_once '../SCFC1/src/LogNotifier.php';
require_once '../SCFC1/src/Personnel.php';
require_once '../SCFC1/src/Player.php';
require_once '../SCFC1/src/Club.php';
require_once '../SCFC1/src/Matches.php';
require_once '../SCFC1/src/Admin.php';
require_once '../SCFC1/src/Member.php';
require_once '../SCFC1/src/SearchLog.php';

// ** Create Objects **//
// User - Member
$user = new User($dbh);
// User - Admin
$admin = new AdminUser($dbh);
// LogSearch
$logSearch = new LogSearch($dbh);
// Fixture
$fixture = new Fixture($dbh);
// Personnel - Inject ClubPersonnel into Dependant Personnel
//$clubPersonnel = new ClubPersonnel($dbh);
$personnel = new Personnel(new ClubPersonnel());
// Players
//$player = new Players();
$players = new Player(new Players());
//Clubs
$club = new Club(new Clubs());
//$clubs = new Clubs();
// Fixtures
//$fixture = new Fixtures();
$match = new Matches(new Fixtures());
// Admins
$admin1 = new Admins();
$admins = new Admin($admin1);
// Members
```

```
// $member = new Members();
$members = new Member(new Members());
// Logs
$log = new SearchLog();
// $logs = new SearchLogs();
// Attach observers
$user->Attach(new LogNotifier());
$logSearch->Attach(new LogNotifier());
```

layout > admintables > admintable.php

```
<? "../SCFC1/config/conninfo.php";

echo '<div class="table-title">
    <h3>Administrators</h3>
</div>';

echo '<div class="add-new">
    <p>Add New</p>
    <a class="nl" onclick="toBottom()" title="Add New Record"></a>
</div>';

?>

<table class="table">
    <thead>
        <tr>
            <th class="text-left">Admin ID</th>
            <th class="text-left">Employee No</th>
            <th class="text-left">Password</th>
            <th class="text-left">Password Hint</th>
            <th class="text-left">Action</th>
        </tr>
    </thead>

    <tbody class="table-hover">
        <?
foreach ($admins->Read($dbh) as $row) {

    echo '<tr><td class="text-left">'.$row["AdminID"].'</td>';
    echo '<td class="text-left">'.$row["EmployeeNo"].'</td>';
    echo '<td class="text-left">'.$row["Password"].'</td>';
    echo '<td class="text-left">'.$row["PasswordHint"].'</td>';
    echo '<td class="text-left">
        <a href="edit.php?AdminID='.$row["AdminID"].'&tableNo=5" class="nl"
title="Edit"></a>

        <a href="delete.php?AdminID='.$row["AdminID"].'&tableNo=5"
class="nl" title="Delete"></a></td></tr>';

}

?>
```

```
</tbody>
</table>
<?
```

layout > admintables > clubtable.php

```
<? include "../SCFC1/config/coninfo.php";

echo '<div class="table-title">
    <h3>Clubs</h3>
</div>';

echo '<div class="add-new">
    <p>Add New</p>
    <a class="nl" onclick="toBottom()" title="Add New Record"></a>
</div>';

?>

<table class="table">
    <thead>
        <tr>
            <th class="text-left">Club ID</th>
            <th class="text-left">Club Name</th>
            <th class="text-left">Club Ground</th>
            <th class="text-left">Address</th>
            <th class="text-left">Postcode</th>
            <th class="text-left">Club Contact</th>
            <th class="text-left">eMail</th>
            <th class="text-left">Phone</th>
            <th class="text-left">Club Badge</th>
            <th class="text-left">Action</th>
        </tr>
    </thead>

    <tbody class="table-hover">

        <?
foreach ($club->Read($dbh) as $row) {

    echo '<tr><td class="text-left">' . $row["ClubID"] . '</td>';
    echo '<td class="text-left">' . $row["ClubName"] . '</td>';
    echo '<td class="text-left">' . $row["ClubGround"] . '</td>';
    echo '<td class="text-left">' . $row["StreetAddress"] . '</td>';
    echo '<td class="text-left">' . $row["Postcode"] . '</td>';
    echo '<td class="text-left">' . $row["ClubContact"] . '</td>';
    echo '<td class="text-left">' . $row["eMail"] . '</td>';
    echo '<td class="text-left">' . $row["PhoneNo"] . '</td>';

    $pic = $row["ClubBadge"];

    if ($pic == "") {

        echo '<td class="text-left"></td>';
    }
    else {

        echo '<td class="text-left"></td>';
```

```

    }

    echo '<td class="text-left">
        <a href="edit.php?ClubID='.$row["ClubID"].'&tableNo=3" class="nl"
        title="Edit"></a>

        <a href="delete.php?ClubID='.$row["ClubID"].'&tableNo=3" class="nl"
        title="Delete"></a></td></tr>';
    }
}

?>

</tbody>
</table>
<?

```

To avoid repetition, only two table includes are provided here as the other tables code is very similar.

layout > create > createadmin.php

```

<form class="form" action="#submitButton" method="post">
<fieldset>

    <div class="personal">
        <h4 class="col-12 newsletter">Add New Record</h4>
        <div class="col-6 entry-box1">
            <input type="text" name="EmployeeNo" placeholder="Employee
Number">
        </div>
        <div class="col-6 entry-box2">
            <input type="text" name="Password" placeholder="Password">
        </div>
        <div class="col-12 entry-box3">
            <input type="text" name="PasswordHint" placeholder="Password
Hint">
        </div>
    </div>

    <input type="submit" name="addNewAdmin" id="submitButton" value="Submit">

</fieldset>
</form>

<div class="news-sign-mess">
<?

    if (isset($_POST["addNewAdmin"])) {

        $addNewAdmin = $_POST["addNewAdmin"];

    }

    if (isset($addNewAdmin)) {

        $error = "";

        if (empty($_POST["EmployeeNo"])) {

```

```

$error = "Employee Number is a Required Field";
?> <div class="error-message"><p><? echo $error ?> </p></div> <?
}
else if (strlen($_POST["EmployeeNo"]) < 4) {

$error = "Employee Number is Too Short";
?> <div class="error-message"><p><? echo $error ?> </p></div> <?
}

else {

include "../SCFC1/config/conninfo.php";

try {

    $admins->eNum = $_POST["EmployeeNo"];
    $admins->pWord = $_POST["Password"];
    $admins->pHint = $_POST["PasswordHint"];

    $admins->Create($dbh, $admins);

} catch (PDOException $e) {

    $e->getMessage();
}

if ($admins->Create($dbh, $admins))
{?<<
<div class="news-success-msg">
    <p>
        <?
        echo "Record added successfully!";
        header ("Refresh:2; admin.php?option=Admins#submitButton");
        ?>
    </p>
</div>
<?}
else
{
    echo "Error!";
}?>

<?

}

}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

?>

</div>
<?

```

layout > create > createstaff.php

```
?>
```

```

        <form class="form" action="#submitButton"
method="post">
            <fieldset>
                <div class="personal">
                    <h4 class="col-12 newsletter">Add New
Record</h4>
                    <div class="col-6 entry-box1">
                        <input type="text" name="FirstName"
placeholder="First Name">
                    </div>
                    <div class="col-6 entry-box2">
                        <input type="text" name="Surname"
placeholder="Surname">
                    </div>
                    <div class="col-6 entry-box1">
                        <input type="text" name="DOB"
placeholder="Date of Birth (YYYY-MM-DD)">
                    </div>
                    <div class="col-6 entry-box2">
                        <input type="text" name="Position"
placeholder="Position">
                    </div>
                    <div class="col-12 entry-box3">
                        <input type="text"
name="StreetAddress" placeholder="Address">
                    </div>
                    <div class="col-6 entry-box1">
                        <input type="text" name="City"
placeholder="City">
                    </div>
                    <div class="col-6 entry-box2">
                        <input type="text" name="Postcode"
placeholder="PostCode">
                    </div>
                    <div class="col-6 entry-box1">
                        <input type="text" name="eMail"
placeholder="eMail">
                    </div>
                    <div class="col-6 entry-box2">
                        <input type="text" name="PhoneNo"
placeholder="Phone">
                    </div>
                </div>
                <input type="submit" name="addNewStaff"
id="submitButton" value="Submit">
            </fieldset>
        </form>
        <div class="news-sign-mess">
            <?
            if (isset($_POST["addNewStaff"])) {
                $addNewStaff = $_POST["addNewStaff"];
            }
            if (isset($addNewStaff)) {
                include "../SCFC1/config/conninfo.php";
                try {

```

```

        $personnel->fName=$_POST["FirstName"];
        $personnel->sName=$_POST["Surname"];
        $personnel->dob=$_POST["DOB"];
        $personnel->position=$_POST["Position"];
        $personnel->sAddress=$_POST["StreetAddress"];
        $personnel->city=$_POST["City"];
        $personnel->pCode=$_POST["Postcode"];
        $personnel->eMail=$_POST["eMail"];
        $personnel->phone=$_POST["PhoneNo"];

        $personnel->Create($dbh, $personnel);
    } catch (PDOException $e) {
        $e->getMessage();
    }

    if ($personnel->Create($dbh, $personnel))
    {
        <div class="news-success-msg">
            <p>
                <?
                echo "Record added
successfully!";
                header ("Refresh:2;
url=admin.php?option=Staff#submitButton");
            </p>
        </div>
        <?>
    } else {
        echo "Error " . $e;
    }
}
?>

</div>
<?

```

Again, to avoid repetition, two files have been provided.

layout > edit > editadmin.php

```

<? if ($table == 5) {
    $aID=$_GET["AdminID"];
    $query=$dbh->prepare("SELECT * FROM admin WHERE AdminID = $aID LIMIT 1");
    $query->execute();
    $aArray = $query->fetchAll(PDO::FETCH_ASSOC);
    ?>
    <div class="table-container">
        <table class="table">
            <thead>
                <tr>
                    <th class="text-left">Admin ID</th>
                    <th class="text-left">Employee No</th>
                    <th class="text-left">Password</th>
                    <th class="text-left">Password Hint</th>

```

```

        <th class="text-left">Action</th>
    </tr>
</thead>

    <tbody class="table-hover">
        <?
foreach ($aArray as $row) {

    echo '<tr><td class="text-left">'.$row["AdminID"].'</td>';
    echo '<td class="text-left">'.$row["EmployeeNo"].'</td>';
    echo '<td class="text-left">'.$row["Password"].'</td>';
    echo '<td class="text-left">'.$row["PasswordHint"].'</td>';
    echo '<td class="text-left">
        <a
href="admin.php?AdminID='.$row["AdminID"].'&tableNo=5" class="nl"
title="Edit"></a>

        <a
href="delete.php?AdminID='.$row["AdminID"].'&tableNo=5" class="nl"
title="Delete"></a></td></tr>';
}

        <?>
    </tbody>
</table>
</div>
<div class="container row">
    <form action="#submitButton" method="post">
        <fieldset>
            <div class="personal">
                <div class="col-6 entry-box4">
                    <input type="text" name="EmployeeNo" placeholder="<?
echo $row["EmployeeNo"]; ?>">
                </div>
                <div class="col-6 entry-box5">
                    <input type="text" name="Password" placeholder="<? echo
$row["Password"]; ?>">
                </div>
                <div class="col-12 entry-box6">
                    <input type="text" name="PasswordHint" placeholder="<?
echo $row["PasswordHint"]; ?>">
                </div>
            </div>

            <input type="submit" name="editAdmin" id="submitButton"
value="Submit">

            </fieldset>
        </form>
        <?
if(isset($_POST["editAdmin"])) {

    try {
        $admins->Update($dbh);
    } catch (PDOException $e) {
        $e->getMessage();
    }
    if ($admins->Update($dbh)) {

        echo "<p>Record Successfully Updated</P>";
        echo "<p class='redirect'>You will be redirected shortly...</p>";
        header("refresh:3; url=admin.php?option=Admins");
    }
    else {
        echo "Error!";

```

```

    }
    ?>
</div>
<?
}
}
}

```

layout > edit > editstaff.php

```

<?
if ($table == 1) {

    $eNum=$_GET["EmployeeNo"];

    $query = $dbh->prepare("SELECT * from clubpersonnel WHERE
EmployeeNo = $eNum LIMIT 1");
    $query->execute();
    $qArray = $query->fetchAll(PDO::FETCH_ASSOC);

    ?>
    <div class="table-container">
        <table class="table">
            <thead>
                <tr>
                    <th class="text-left">Employee No.</th>
                    <th class="text-left">First Name</th>
                    <th class="text-left">Surname</th>
                    <th class="text-left">DOB</th>
                    <th class="text-left">Position</th>
                    <th class="text-left">Address</th>
                    <th class="text-left">City</th>
                    <th class="text-left">Postcode</th>
                    <th class="text-left">eMail</th>
                    <th class="text-left">Phone</th>
                    <th class="text-left">Action</th>
                </tr>
            </thead>
            <tbody class="table-hover">
                <?
                foreach ($qArray as $row) {

                    echo '<tr><td class="text-
left">'.$row["EmployeeNo"].'</td>';
                    echo '<td class="text-left">'.$row["FirstName"].'</td>';
                    echo '<td class="text-left">'.$row["Surname"].'</td>';
                    echo '<td class="text-left">'.$row["DOB"].'</td>';
                    echo '<td class="text-left">'.$row["Position"].'</td>';
                    echo '<td class="text-left">'.$row["StreetAddress"].'</td>';
                    echo '<td class="text-left">'.$row["City"].'</td>';
                    echo '<td class="text-left">'.$row["Postcode"].'</td>';
                    echo '<td class="text-left">'.$row["eMail"].'</td>';
                    echo '<td class="text-left">'.$row["PhoneNo"].'</td>';
                    echo '<td class="text-left">
                        <a
href="edit.php?EmployeeNo='.$row["EmployeeNo"].'&tableNo=1" class="nl"
title="Edit"></a>

                        <a
href="delete.php?EmployeeNo='.$row["EmployeeNo"].'&tableNo=1" class="nl"
title="Delete"></a></td></tr>';

```

```

    }
    ?>

        </tbody>
    </table>
</div>
<div class="container row">
    <form action="#submitButton" method="post">
        <fieldset>

            <div class="personal">
                <div class="col-6 entry-box4">
                    <input type="text" name="FirstName"
value="<? echo $row["FirstName"]; ?>">
                </div>
                <div class="col-6 entry-box5">
                    <input type="text" name="Surname" value="<?
echo $row["Surname"]; ?>">
                </div>
                <div class="col-6 entry-box4">
                    <input type="text" name="DOB"
placeholder="<? echo $row["DOB"]; ?>">
                </div>
                <div class="col-6 entry-box5">
                    <input type="text" name="Position"
placeholder="<? echo $row["Position"]; ?>">
                </div>
                <div class="col-12 entry-box6">
                    <input type="text" name="StreetAddress"
placeholder="<? echo $row["StreetAddress"]; ?>">
                </div>
                <div class="col-6 entry-box4">
                    <input type="text" name="City"
placeholder="<? echo $row["City"]; ?>">
                </div>
                <div class="col-6 entry-box5">
                    <input type="text" name="Postcode"
placeholder="<? echo $row["Postcode"]; ?>">
                </div>
                <div class="col-6 entry-box4">
                    <input type="text" name="eMail"
placeholder="<? echo $row["eMail"]; ?>">
                </div>
                <div class="col-6 entry-box5">
                    <input type="text" name="PhoneNo"
placeholder="<? echo $row["PhoneNo"]; ?>">
                </div>
            </div>

            <input type="submit" name="editStaff"
id="submitButton1" value="Submit">

        </fieldset>
    </form>
    <?
    if (isset($_POST["editStaff"])) {

        try {
            $personnel->Update($dbh);
        } catch (PDOException $e) {
            $e->getMessage();
        }

        if ($personnel->Update($dbh)) {

```

```

        echo "<p>Record Successfully Updated</P>";
        echo "<p class='redirect'>You will be redirected
shortly...</p>";
        header("refresh:3;
url=admin.php?option=Staff#".$row["EmployeeNo"]."""); // Doesnt Work?!
    }
    else {
        echo "Error: Record Not Updated!";
    }
}
?>
</div>
<?
}
}
?>

```

Again, two files shown.

layout > header.php

```

<header id="header">
    <div class="container row">
        <div class="col-1 logo">
            <a href="../SCFC1/index.php"></a>
        </div>

        <div class="user-name">
            <?
                require_once "../SCFC1/config/conninfo.php";

                session_start();

                // Is user logged in
                if ($user->LoggedIn()) {

                    echo "Hello, " . $_SESSION["userLoggedIn"];
                    // TODO: Get First Name from array to display.
                    // foreach ($user->GetName($dbh) as $row)
                    // {
                    //     echo "Hello, " . $row["FirstName"];
                    // }

                }

                if ($admin->LoggedIn()) {

                    foreach ($admin->GetName($dbh) as $row)
                    {
                        echo "Hello, " . $row["FirstName"];
                    }

                }

            ?>

        </div>

    <?

    if ($user->LoggedIn() || $admin->LoggedIn())
    {
        ?>
    }

```

```

        <div class="col-11 icons">
            <a href="../SCFC1/logout.php"></a>
            <a href="../stadium.php"></a>
        </div>

        <nav class="col-11 main-nav">
            <ul class="menu">
                <li><a href="../SCFC1/index.php">Home</a></li>
                <li><a href="../SCFC1/news.php">News</a></li>
                <li><a href="../SCFC1/match.php">Match</a></li>
                <li><a href="../SCFC1/team.php">Team</a></li>
                <? if ($admin->LoggedIn())
                    {
                        <? <li><a href="admin.php">Admin</a></li> <?
                    }
                <?
            </ul>
        </nav>

        <?
    } else {
        <?

        <div class="col-11 icons">
            <a href="../SCFC1/adminlogin.php"></a>
            <a href="../SCFC1/login.php"></a>
            <a class="nl" href="../SCFC1/register.php"
title="Register"></a>
            <a href="../SCFC1/stadium.php"></a>
        </div>

        <nav class="col-11 main-nav">
            <ul class="menu">
                <li><a href="../SCFC1/index.php">Home</a></li>
                <li><a href="../SCFC1/news.php">News</a></li>
            </ul>
        </nav>

        <?
    }
    <?
</div>
</header>

```

```

<nav class="col-11 main-nav">
    <ul class="menu">
        <li><a href="../SCFC1/index.php">Home</a></li>
        <li><a href="../SCFC1/news.php">News</a></li>
    </ul>
</nav>

<?

```

```

    }

    ?>
</div>
</header>

```

layout > signup.php

```

<div class="container row">
    <form class="form" action="#submitButton" method="post">
        <fieldset>
            <div class="personal">
                <h4 class="col-12 newsletter">register for scfc archive</h4>
                <div class="col-6 entry-box1">
                    <input type="text" name="FirstName" placeholder="First
Name">
                </div>
                <div class="col-6 entry-box2">
                    <input type="text" name="Surname" placeholder="Last
Name">
                </div>
                <div class="col-12 entry-box3">
                    <input type="text" name="eMail" placeholder="eMail">
                </div>
                <div class="col-6 entry-box2" style="display: none;">
                    <input type="text" name="spam" type="hidden" value=""/>
                </div>
                <div class="col-6 entry-box1">
                    <input type="password" name="password"
placeholder="Choose a Password">
                </div>
                <div class="col-6 entry-box2">
                    <input type="password" name="vPassword"
placeholder="Verify Password">
                </div>
            </div>
            <input type="submit" name="register" id="submitButton"
value="Submit">
        </fieldset>
    </form>
    <div class="news-sign-mess">
        <?
            require_once "../SCFC1/config/conninfo.php";

            // Is User Logged In?
            if ($user->LoggedIn()) {
                $user->Redirect();
            }

            if (@$_POST["spam"] != '') {
                // Do nothing //
            }
            else {

                if (isset($_POST["register"])) {

                    $newsSubmit = $_POST["register"];

                }

                if (isset($newsSubmit)) {

```

```

        $error = $eError = $sfnError = $ssnError = "";
        $fName = test_input($_POST["FirstName"]);
        $sName = test_input($_POST["Surname"]);
        $eMail = test_input($_POST["eMail"]);
        $password = test_input($_POST["password"]);
        $vPassword = test_input($_POST["vPassword"]);

        if (empty($_POST["FirstName"])) {

            $error = "First Name is a Required Field";
            ?> <div class="error-message"><p><? echo $error ?>
</p></div> <?
        }
        else if (empty($_POST["Surname"])) {

            $error = "Surname is a Required Field";
            ?> <div class="error-message"><p><? echo $error ?>
</p></div> <?
        }
        else if (empty($_POST["eMail"])) {

            $error = "eMail is a Required Field";
            ?> <div class="error-message"><p><? echo $error ?>
</p></div> <?
        }
        else if (empty($_POST["password"])) {

            $error = "Password is a Required Field";
            ?> <div class="error-message"><p><? echo $error ?>
</p></div> <?
        }
        else if (empty($_POST["vPassword"])) {

            $error = "Please Verifiy your Password";
            ?> <div class="error-message"><p><? echo $error ?>
</p></div> <?
        }
        else if ($_POST["password"] != $_POST["vPassword"]) {

            $error = "Passwords do not Match";
            ?> <div class="error-message"><p><? echo $error ?>
</p></div> <?
        }
        else if (!preg_match("/^[a-zA-Z ]*$/", $fName)) {

            $sfnError = "No Special Characters in Name";
            ?> <div class="error-message"><p><? echo $sfnError
?> </p></div> <?
        }
        else if (!preg_match("/^[a-zA-Z ]*$/", $sName)) {

            $ssnError = "No Special Characters in Surname";
            ?> <div class="error-message"><p><? echo $ssnError
?> </p></div> <?
        }
        else if (!filter_var($eMail, FILTER_VALIDATE_EMAIL)) {

            $eError = "Invalid email format";
            ?> <div class="error-message"><p><? echo $eError ?>
</p></div> <?
        }

    }
    else {

```

```

        try {
            if ($user->RegisteredCheck($dbh, $eMail) ==
true)
            {
                ?>
                <div class="error-message">
                <p>
                <? echo "e-Mail Already Registered";
?>
                </p>
                </div>
                <?
            }
            else
            {
                $user->Register($dbh, $fName, $sName,
$eMail, $password);
                //header ('Refresh:5;
url=../SCFC1/login.php');
            }

        } catch (PDOException $e) {

            echo $e->getMessage();

        }

        require_once '../SCFC1/src/interfaces/IObserver.php';
        require_once '../SCFC1/src/LogNotifier.php';

    }

}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

?>

</div>

<? include 'layout/sponsors.php'; ?>

</div>

```

src > interfaces > INewUserObserver.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 14/01/2019
 * Time: 22:12
 */

require_once '../SCFC1/src/interfaces/IObserver.php';

```

```

interface INewUserObserver
{
    public function Attach($newUser);
    public function Detach($newUser);
    public function Notify($eMail);
}

```

src > interfaces > IObserver.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 14/01/2019
 * Time: 22:11
 */

require_once '../SCFC1/src/interfaces/INewUserObserver.php';

interface IObserver
{
    public function NewUserReg($newUser, $data);

    public function SearchLog($dbh, $userEmail, $keyword, $filter,
$dateTime);
}

```

src > interfaces > ISearchObserver.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 10:35
 */

require_once '../SCFC1/src/interfaces/IObserver.php';

interface ISearchObserver
{
    public function Attach($newSearchLog);
    public function Detach($newSearchLog);
    public function Notify($userEmail, $keyword, $filter, $dateTime);
}

```

src > interfaces > IRepositoryInterface.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 16/01/2019
 * Time: 22:05
 */

interface IRepositoryInterface
{
    public function Create($dbh, $data);

    public function Read($dbh);
}

```

```
    public function Update($dbh);  
    public function Delete($dbh);  
}
```

src > models > Admins.php

```
<?php  
/**  
 * Created by PhpStorm.  
 * User: Peter  
 * Date: 15/01/2019  
 * Time: 13:43  
 */  
  
class Admins  
{  
    public $adminId;  
    public $eNo;  
    public $password;  
    public $pHint;  
}
```

src > models > ClubPersonnel.php

```
<?php  
  
class ClubPersonnel  
{  
    public $employeeNo;  
    public $firstName;  
    public $surname;  
    public $DOB;  
    public $position;  
    public $streetAddress;  
    public $city;  
    public $postcode;  
    public $eMail;  
    public $phoneNo;  
}
```

Two models shown.

src > Admin.php

```
<?php  
/**  
 * Created by PhpStorm.  
 * User: Peter  
 * Date: 15/01/2019  
 * Time: 20:24  
 */
```

```
require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once '../SCFC1/src/models/Admins.php';
require_once '../SCFC1/config/conninfo.php';

class Admin implements IRepositoryInterface
{
    public function Create($dbh, $data)
    {
        try {
            // if(is_a($data, "Admins")){
            //     return false;
            // }

            // $eNum=$_POST["EmployeeNo"];
            // $password=$_POST["Password"];
            // $pHint=$_POST["PasswordHint"];

            $eNum = $data->eNum;
            $password = $data->pWord;
            $pHint = $data->pHint;

            $insert = $dbh->prepare("INSERT INTO admin(EmployeeNo, Password, PasswordHint) VALUES (:EmployeeNo, :Password, :PasswordHint)");
            $insert->bindParam(':EmployeeNo', $eNum);
            $insert->bindParam(':Password', $password);
            $insert->bindParam('PasswordHint', $pHint);
            $insert->execute();
            return true;
        } catch (PDOException $e) {
            throw $e;
        }
    }

    public function Read($dbh)
    {
        try {
            $getAdmin = $dbh->prepare("SELECT * FROM admin");
            $getAdmin->execute();
            $adminArray = $getAdmin->fetchAll(PDO::FETCH_ASSOC);
            return $adminArray;
        } catch (PDOException $e) {
            {
                $e->getMessage();
            }
        }
    }

    public function Update($dbh)
    {
        try {
            $aID=$_GET["AdminID"];
            $eNum=$_POST["EmployeeNo"];
            $password=$_POST["Password"];
            $pHint=$_POST["PasswordHint"];

            $insert = $dbh->prepare("UPDATE admin
                                SET EmployeeNo = IF('$eNum' = '',
EmployeeNo, '$eNum'),
                                Password = IF('$password' = '',
Password, '$password')");
            $insert->execute();
        } catch (PDOException $e) {
            throw $e;
        }
    }
}
```

```

Password, '$password'),
                                PasswordHint = IF('$pHint' = '',
PasswordHint, '$pHint') WHERE AdminID = $aID LIMIT 1");

        $insert->execute();
        return true;
    } catch (PDOException $e) {
        throw $e;
    }
}

public function Delete($dbh)
{
    $aID=$_GET["AdminID"];
    try {
        $query = $dbh->prepare("DELETE FROM admin WHERE AdminID = $aID
LIMIT 1");
        $query->execute();
    } catch (PDOException $e) {
        $e->getMessage();
    }
}
}

```

src > AdminUser.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 16:11
 */

require_once "../SCFC1/config/conninfo.php";

class AdminUser
{
    // Attributes

    // Properties

    // Methods

    // Register new Members

    // Log Users In
    public function Login($dbh, $eNumber, $password)
    {
        try {
            // Get all users from members table
            $admin = $dbh->prepare("SELECT * FROM admin WHERE
EmployeeNo=:EmployeeNo LIMIT 1");
            $admin->bindParam(":EmployeeNo", $eNumber);
            $admin->execute();

            // Returned row as array indexed by column name
            $result = $admin->fetch(PDO::FETCH_ASSOC);

            // Check if row has been returned
            if ($admin->rowCount() > 0)
            {
                if ($password == $result["Password"])

```

```

        {
            $_SESSION["adminLoggedIn"] = $result["EmployeeNo"];
            return true;
        }
        else
        {
            return false;
            echo 'Incorrect Username or Password';
        }
    }
}

}
catch (PDOException $e)
{
    echo $e->getMessage();
}
}

// TODO: GetFirstName Something wrong with query? invalid argument for
foreach
public function GetName($dbh)
{
    $eNum = $_SESSION["adminLoggedIn"];
    try {
        $getName = $dbh->prepare("SELECT * from clubpersonnel, admin
WHERE clubpersonnel.EmployeeNo = admin.EmployeeNo and
clubpersonnel.EmployeeNo = $eNum");
        $getName->execute();
        $nameMatch = $getName->fetchAll(PDO::FETCH_ASSOC);
        return $nameMatch;
    }
    catch (PDOException $e) {
        echo $e->getMessage();
    }
}

// Redirect Users
public function Redirect()
{
    header('Location: ../SCFC1/admin.php');
}

// Check if User is Logged in
public function LoggedIn()
{
    if (isset($_SESSION["adminLoggedIn"]) && $_SESSION["adminLoggedIn"]
!= "")
    {
        return true;
    }
}

public function TimeOut()
{
    if (isset($_SESSION["LastActivity"]) && (time() -
$_SESSION["LastActivity"] > 1300)) {
        session_start();
        unset($_SESSION["adminLoggedIn"]);
        session_destroy();
    }
}

```

```

        echo "Logged out. Redirecting...";
        header ("Location: ../SCFC1/timeout.php");
        return true;
    }
    $_SESSION["LastActivity"] = time();
}

// Log Out
public function Logout()
{
    session_start();
    unset($_SESSION["adminLoggedIn"]);
    session_destroy();
    echo "Logged out. Redirecting...";
    header( "refresh:2;url=../SCFC1/index.php" );
    return true;
}
}
?>

```

src > Club.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 20:52
 */

require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once "../SCFC1/src/models/Clubs.php";
require_once "../SCFC1/config/conninfo.php";

class Club implements IRepositoryInterface
{
    // Methods
    public function Create($dbh, $data)
    {
        try {
            $cName=$data->cName;
            $cGround=$data->cGround;
            $sAddress=$data->sAddress;
            $pCode=$data->pCode;
            $cCon=$data->cCon;
            $eMail=$data->eMail;
            $phone=$data->phone;
            $cBadge=$data->cBadge;

            $insert = $dbh->prepare("INSERT INTO clubs(ClubName, ClubGround,
StreetAddress, PostCode, ClubContact, eMail, PhoneNo, ClubBadge)
VALUES (:ClubName, :ClubGround,
:StreetAddress, :PostCode, :ClubContact, :eMail, :PhoneNo, :ClubBadge)");
            $insert->bindParam(':ClubName', $cName);
            $insert->bindParam(':ClubGround', $cGround);
            $insert->bindParam(':StreetAddress', $sAddress);
            $insert->bindParam(':Postcode', $pCode);
            $insert->bindParam(':ClubContact', $cCon);

```

```

        $insert->bindParam(':eMail', $eMail);
        $insert->bindParam(':PhoneNo', $phone);
        $insert->bindParam(':ClubBadge', $cBadge);
        $insert->execute();
        return true;
    } catch (PDOException $e) {
        throw $e;
    }
}

public function Read($dbh)
{
    try {
        $getClubs = $dbh->prepare("SELECT * FROM clubs");
        $getClubs->execute();
        $clubArray = $getClubs->fetchAll(PDO::FETCH_ASSOC);
        return $clubArray;
    } catch (PDOException $e)
    {
        $e->getMessage();
    }
}

public function Update($dbh)
{
    try {
        $cID=$_GET["ClubID"];
        $cName=$_POST["ClubName"];
        $cGround=$_POST["ClubGround"];
        $sAddress=$_POST["StreetAddress"];
        $pCode=$_POST["Postcode"];
        $cCon=$_POST["ClubContact"];
        $eMail=$_POST["eMail"];
        $phone=$_POST["PhoneNo"];
        $cBadge=$_POST["ClubBadge"];

        $insert = $dbh->prepare("UPDATE clubs
SET ClubName = IF('$cName' = '', ClubName, '$cName'),
ClubGround = IF('$cGround' = '', ClubGround, '$cGround'),
StreetAddress = IF('$sAddress' = '', StreetAddress,
'$sAddress'),
PostCode = IF('$pCode' = '', PostCode, '$pCode'),
ClubContact = IF('$cCon' = '', ClubContact, '$cCon'),
eMail = IF('$eMail' = '', eMail, '$eMail'),
PhoneNo = IF('$phone' = '', PhoneNo, '$phone'),
ClubBadge = IF('$cBadge' = '', ClubBadge, '$cBadge') WHERE
ClubID = $cID LIMIT 1");

        $insert->execute();
        return true;
    } catch (PDOException $e) {
        throw $e;
    }
}

public function Delete($dbh)
{
    $cID=$_GET["ClubID"];
    try {
        $query = $dbh->prepare("DELETE FROM clubs WHERE ClubID = $cID

```

```

LIMIT 1");
        $query->execute();
    } catch (PDOException $e) {
        $e->getMessage();
    }
}
}
}

```

src > Club.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 20:52
 */

require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once '../SCFC1/src/models/Clubs.php';
require_once '../SCFC1/config/conninfo.php';

class Club implements IRepositoryInterface
{
    // Methods
    public function Create($dbh, $data)
    {
        try {
            $cName=$data->cName;
            $cGround=$data->cGround;
            $sAddress=$data->sAddress;
            $pCode=$data->pCode;
            $cCon=$data->cCon;
            $eMail=$data->eMail;
            $phone=$data->phone;
            $cBadge=$data->cBadge;

            $insert = $dbh->prepare("INSERT INTO clubs(ClubName, ClubGround,
StreetAddress, PostCode, ClubContact, eMail, PhoneNo, ClubBadge)
VALUES (:ClubName, :ClubGround,
:StreetAddress, :PostCode, :ClubContact, :eMail, :PhoneNo, :ClubBadge)");
            $insert->bindParam(':ClubName', $cName);
            $insert->bindParam(':ClubGround', $cGround);
            $insert->bindParam(':StreetAddress', $sAddress);
            $insert->bindParam(':Postcode', $pCode);
            $insert->bindParam(':ClubContact', $cCon);
            $insert->bindParam(':eMail', $eMail);
            $insert->bindParam(':PhoneNo', $phone);
            $insert->bindParam(':ClubBadge', $cBadge);
            $insert->execute();
            return true;
        } catch (PDOException $e) {
            throw $e;
        }
    }

    public function Read($dbh)
    {
        try {

```

```

        $getClubs = $dbh->prepare("SELECT * FROM clubs");
        $getClubs->execute();
        $clubArray = $getClubs->fetchAll(PDO::FETCH_ASSOC);
        return $clubArray;
    } catch (PDOException $e)
    {
        $e->getMessage();
    }
}

public function Update($dbh)
{
    try {
        $cID=$_GET["ClubID"];
        $cName=$_POST["ClubName"];
        $cGround=$_POST["ClubGround"];
        $sAddress=$_POST["StreetAddress"];
        $pCode=$_POST["Postcode"];
        $cCon=$_POST["ClubContact"];
        $eMail=$_POST["eMail"];
        $phone=$_POST["PhoneNo"];
        $cBadge=$_POST["ClubBadge"];

        $insert = $dbh->prepare("UPDATE clubs
SET ClubName = IF('$cName' = '', ClubName, '$cName'),
ClubGround = IF('$cGround' = '', ClubGround, '$cGround'),
'$sAddress'),
PostCode = IF('$pCode' = '', PostCode, '$pCode'),
ClubContact = IF('$cCon' = '', ClubContact, '$cCon'),
eMail = IF('$eMail' = '', eMail, '$eMail'),
PhoneNo = IF('$phone' = '', PhoneNo, '$phone'),
ClubBadge = IF('$cBadge' = '', ClubBadge, '$cBadge') WHERE
ClubID = $cID LIMIT 1");

        $insert->execute();
        return true;
    } catch (PDOException $e) {
        throw $e;
    }
}

public function Delete($dbh)
{
    $cID=$_GET["ClubID"];
    try {
        $query = $dbh->prepare("DELETE FROM clubs WHERE ClubID = $cID
LIMIT 1");
        $query->execute();
    } catch (PDOException $e) {
        $e->getMessage();
    }
}
}

```

src >Fixture.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019

```

```

* Time: 11:48
*/

require_once "../SCFC1/config/conninfo.php";

class Fixture
{
    // Attributes

    // Constructor

    // Properties

    // Methods
    public function GetFixtures($dbh)
    {
        try {
            $getFixtures = $dbh->prepare("SELECT * FROM fixtures, clubs
WHERE fixtures.ClubID = clubs.ClubID");
            $getFixtures->execute();
            $numRows = $getFixtures->rowCount();
            return $numRows;

        } catch (PDOException $e) {

            echo $e->getMessage();

        }

    }

    public function PageFixtures($dbh)
    {
        $limit = 3;

        if (empty($page)) {
            $page = 0;
        }

        try {
            $pagedFixtures = $dbh->prepare("SELECT * FROM fixtures, clubs
WHERE fixtures.ClubID = clubs.ClubID
LIMIT $page, $limit");
            $pagedFixtures->execute();

            $count = 1 + $page;

            return $pagedFixtures;

        } catch (PDOException $e) {
            $e->getMessage();
        }

    }
}

```

src >LogNotifier.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 14/01/2019
 * Time: 22:38

```

```

*/

require_once '../SCFC1/src/interfaces/IObserver.php';
require_once '../SCFC1/config/conninfo.php';

class LogNotifier implements IObserver
{
    // Attributes
    // Constructor
    // Properties

    // Methods
    public function NewUserReg($newUser, $data)
    {
        // TODO: Get mail server working through WAMP
        // $mail = mail("pdavies.079868@student.uwtsd.ac.uk", "New SCFC
        Archive Member!", "A new user with email " . $data . "has registered!");
        ?><div class="error-message"><p><? echo 'New User ' . $data . '
        Added!'; ?></p></div><?
    }

    public function SearchLog($dbh, $userEmail, $keyword, $filter,
    $dateTime)
    {
        echo "Logged";
    }
}

```

src >LogSearch.php

```

<?php

require_once '../SCFC1/src/interfaces/ISearchObserver.php';
require_once '../SCFC1/src/interfaces/IObserver.php';
require_once '../SCFC1/config/conninfo.php';

class LogSearch implements ISearchObserver
{
    // Attributes
    // Constructor
    // Properties

    // Methods
    public function Search($dbh, $filter, $keyword)
    {
        try {
            $searchPlayers = $dbh->prepare("SELECT * FROM clubpersonnel, players
            WHERE clubpersonnel.EmployeeNO =
            players.EmployeeNO AND $filter like '$keyword'");
            $searchPlayers->execute();
            return $searchPlayers->fetchAll(PDO::FETCH_ASSOC);
        } catch (PDOException $e) {
            echo $e->getMessage();
        }
    }

    public function SearchLog($dbh, $userEmail, $keyword, $filter,

```

```

$dateTime)
    {
        try {
            $searchLog = $dbh->prepare("INSERT INTO searchlogs (userEmail,
keyword, filter, dateTime)
                                VALUES (:userEmail, :keyword, :filter,
:dateTime)");
            $searchLog->bindParam(':userEmail', $userEmail);
            $searchLog->bindParam(':keyword', $keyword);
            $searchLog->bindParam(':filter', $filter);
            $searchLog->bindParam(':dateTime', $dateTime);
            $searchLog->execute();

        } catch (PDOException $e) {

            echo $e->getMessage();

        }
    }

    // Observer code
    private $_newSearchLog = array();

    public function Attach($newSearchLog)
    {
        $this->_newSearchLog = $newSearchLog;
    }

    public function Detach($newSearchLog)
    {
        // TODO: Implement Detach() method.
    }

    public function Notify($userEmail, $keyword, $filter, $dateTime)
    {
        foreach ($this->_newSearchLog as $sLog)
        {
            $sLog->SearchLog($this, $userEmail, $keyword, $filter,
$dateTime);
        }
    }
}

```

src >Matches.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 20:24
 */

require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once "../SCFC1/src/models/Fixtures.php";
require_once "../SCFC1/config/conninfo.php";

class Matches implements IRepositoryInterface
{

    public function Create($dbh, $data)
    {
        try {

            $cID = $data->cID;
            $date = $data->date;

```

```

        $time = $data->time;
        $result = $data->result;

        $insert = $dbh->prepare("INSERT INTO fixtures(ClubID, Date,
Time, Result) VALUES (:ClubID, :Date, :Time, :Photo)");
        $insert->bindParam(':ClubID', $cID);
        $insert->bindParam(':Date', $date);
        $insert->bindParam(':Time', $time);
        $insert->bindParam(':Result', $result);
        $insert->execute();
        return true;

    } catch (PDOException $e) {
        throw $e;
    }
}

public function Read($dbh)
{
    try {

        $getFix = $dbh->prepare("SELECT * FROM fixtures");
        $getFix->execute();
        $fixArray = $getFix->fetchAll(PDO::FETCH_ASSOC);
        return $fixArray;

    } catch (PDOException $e)
    {
        $e->getMessage();
    }
}

public function Update($dbh)
{
    try {

        $fID=$_GET["FixtureID"];
        $cID=$_POST["ClubID"];
        $date=$_POST["Date"];
        $time=$_POST["Time"];
        $result=$_POST["Result"];

        $insert = $dbh->prepare("UPDATE fixtures
SET ClubID = IF('$cID' = '', ClubID, '$cID'),
    Date = IF('$date' = '', Date, '$date'),
    Time = IF('$time' = '', Time, '$time'),
    Result = IF('$result' = '', Result, '$result') WHERE
FixtureID = $fID LIMIT 1");

        $insert->execute();
        return true;

    } catch (PDOException $e) {
        throw $e;
    }
}

public function Delete($dbh)
{
    $fID=$_GET["FixtureID"];
    try {
        $query = $dbh->prepare("DELETE FROM fixtures WHERE FixtureID =
$fID LIMIT 1");
        $query->execute();
    } catch (PDOException $e) {
        $e->getMessage();
    }
}

```

```

    }
}
}

```

src >Member.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 20:24
 */

require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once '../SCFC1/src/models/Members.php';
require_once '../SCFC1/config/conninfo.php';

class Member implements IRepositoryInterface
{
    // Methods
    public function Create($dbh, $data)
    {
        try {

            $fName = $data->fName;
            $sName = $data->sName;
            $eMail = $data->eMail;
            $hashPassword = $data->hashPassword;

            $insert = $dbh->prepare("INSERT INTO members(FirstName, Surname,
eMail, password) VALUES (:FirstName, :Surname, :eMail, :password)");

            $insert->bindParam(':FirstName', $fName);
            $insert->bindParam(':Surname', $sName);
            $insert->bindParam(':eMail', $eMail);
            $insert->bindParam(':password', $hashPassword);
            $insert->execute();

        }
        catch (PDOException $e) {
            throw $e;
        }
    }

    public function Read($dbh)
    {
        try {

            $getMember = $dbh->prepare("SELECT * FROM members");
            $getMember->execute();
            $memArray = $getMember->fetchAll(PDO::FETCH_ASSOC);
            return $memArray;

        } catch (PDOException $e)
        {
            $e->getMessage();
        }
    }

    public function Update($dbh)
    {

```

```

        try {
            $mID=$_GET["MListID"];
            $fName=$_POST["FirstName"];
            $sName=$_POST["Surname"];
            $eMail=$_POST["eMail"];

            $insert = $dbh->prepare("UPDATE mailinglist
            SET eMail = IF('$eMail' = '', eMail, '$eMail'),
                FirstName = IF('$fName' = '', FirstName, '$fName'),
                Surname = IF('$sName' = '', Surname, '$sName') WHERE MListID
            = $mID LIMIT 1");

            $insert->execute();
            return true;
        } catch (PDOException $e) {
            throw $e;
        }
    }

    public function Delete($dbh)
    {
        $mID=$_GET["MemID"];
        try {
            $query = $dbh->prepare("DELETE FROM members WHERE MemID = $mID
            LIMIT 1");
            $query->execute();
        } catch (PDOException $e) {
            $e->getMessage();
        }
    }
}

```

src >Personnel.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 14:37
 */

require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once "../SCFC1/src/models/ClubPersonnel.php";
require_once "../SCFC1/config/conninfo.php";

class Personnel implements IRepositoryInterface
{
    public function Create($dbh, $data)
    {
        try {
            $fName=$data->fName;
            $sName=$data->sName;
            $dob=$data->dob;
            $position=$data->position;
            $sAddress=$data->sAddress;
            $city=$data->city;
            $pCode=$data->pCode;
            $eMail=$data->eMail;
            $phone=$data->phone;

```

```

        $insert = $dbh->prepare("INSERT INTO clubpersonnel(FirstName,
Surname, DOB, Position, StreetAddress, City, Postcode, eMail, PhoneNo)
VALUES (:FirstName, :Surname, :DOB, :Position, :StreetAddress, :City,
:Postcode, :eMail, :PhoneNo)");
        $insert->bindParam(':FirstName', $fName);
        $insert->bindParam(':Surname', $sName);
        $insert->bindParam('DOB', $dob);
        $insert->bindParam(':Position', $position);
        $insert->bindParam(':StreetAddress', $sAddress);
        $insert->bindParam(':City', $city);
        $insert->bindParam(':Postcode', $pCode);
        $insert->bindParam('eMail', $eMail);
        $insert->bindParam('PhoneNo', $phone);
        $insert->execute();
        return true;
    } catch (PDOException $e) {
        throw $e;
    }
}

public function Read($dbh)
{
    try {

        $getStaff = $dbh->prepare("SELECT * FROM clubpersonnel");
        $getStaff->execute();
        $staffArray = $getStaff->fetchAll(PDO::FETCH_ASSOC);
        return $staffArray;

    } catch (PDOException $e)
    {
        $e->getMessage();
    }
}

public function Update($dbh)
{
    try {

        $eNum=$_GET["EmployeeNo"];

        $fName=$_POST["FirstName"];
        $sName=$_POST["Surname"];
        $dob=$_POST["DOB"];
        $position=$_POST["Position"];
        $sAddress=$_POST["StreetAddress"];
        $city=$_POST["City"];
        $pCode=$_POST["Postcode"];
        $eMail=$_POST["eMail"];
        $phone=$_POST["PhoneNo"];

        $insert = $dbh->prepare("UPDATE clubpersonnel
SET FirstName = IF('$fName' = '', FirstName, '$fName'),
Surname = IF('$sName' = '', Surname, '$sName'),
DOB = IF('$dob' = '', DOB, '$dob'),
Position = IF('$position' = '', Position, '$position'),
StreetAddress = IF('$sAddress' = '', StreetAddress,
'$sAddress'),
City = IF('$city' = '', City, '$city'),
Postcode = IF('$pCode' = '', Postcode, '$pCode'),
eMail = IF('$eMail' = '', eMail, '$eMail'),
PhoneNo = IF('$phone' = '', PhoneNo, '$phone') WHERE EmployeeNo
= $eNum LIMIT 1");

```

```

        $insert->execute();
        return true;
    } catch (PDOException $e) {
        throw $e;
    }
}

public function Delete($dbh)
{
    $eNum=$_GET["EmployeeNo"];
    try {
        $query = $dbh->prepare("DELETE FROM clubpersonnel WHERE
EmployeeNo = $eNum LIMIT 1");
        $query->execute();
    } catch (PDOException $e) {
        $e->getMessage();
    }
}
}
}

```

src >Player.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 20:24
 */

require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once "../SCFC1/src/models/Players.php";
require_once "../SCFC1/config/conninfo.php";

class Player implements IRepositoryInterface
{
    // Methods
    public function Create($dbh, $data)
    {
        try {

            $eNum = $data->eNum;
            $pPosition=$data->pPosition;
            $pNum=$data->pNum;
            $photo=$data->photo;

            $insert = $dbh->prepare("INSERT INTO players(EmployeeNo,
PlayingPosition, PlayerNo, Photo) VALUES (:EmployeeNo, :PlayingPosition,
:PlayerNo, :Photo)");
            $insert->bindParam(':EmployeeNo', $eNum);
            $insert->bindParam(':PlayingPosition', $pPosition);
            $insert->bindParam('PlayerNo', $pNum);
            $insert->bindParam(':Photo', $photo);
            $insert->execute();
            return true;

        } catch (PDOException $e) {
            throw $e;
        }
    }
}

```

```

public function Read($dbh )
{
    try {

        $getPlayers = $dbh->prepare("SELECT * FROM players");
        $getPlayers->execute();
        $playerArray = $getPlayers->fetchAll(PDO::FETCH_ASSOC);
        return $playerArray;

    } catch (PDOException $e)
    {
        $e->getMessage();
    }
}

public function Update($dbh)
{
    try {

        $pID=$_GET["PlayerID"];
        $eNum=$_POST["EmployeeNo"];
        $pPosition=$_POST["PlayingPosition"];
        $pNum=$_POST["PlayerNo"];
        $photo=$_POST["Photo"];

        $insert = $dbh->prepare("UPDATE players
SET EmployeeNo = IF('$eNum' = '', EmployeeNo, '$eNum'),
PlayingPosition = IF('$pPosition' = '', PlayingPosition,
'$pPosition'),
PlayerNo = IF('$pNum' = '', PlayerNo, '$pNum'),
Photo = IF('$photo' = '', Photo, '$photo') WHERE PlayerID = $pID
LIMIT 1");

        $insert->execute();
        return true;

    } catch (PDOException $e) {
        throw $e;
    }
}

public function Delete($dbh)
{
    $pID=$_GET["PlayerID"];
    try {
        $query = $dbh->prepare("DELETE FROM players WHERE PlayerID =
$pID LIMIT 1");
        $query->execute();
    } catch (PDOException $e) {
        $e->getMessage();
    }
}
}

```

src >SearchLog.php

```

<?php
/**
 * Created by PhpStorm.
 * User: Peter
 * Date: 15/01/2019
 * Time: 20:24
 */

```

```

require_once '../SCFC1/src/interfaces/IRepositoryInterface.php';
require_once "../SCFC1/src/models/SearchLogs.php";
require_once "../SCFC1/config/conninfo.php";

class SearchLog //implements IRepositoryInterface
{
    // Methods

    public function Create($dbh)
    {
        // TODO: No need for user to create logs // part of observer
        pattern.
    }

    public function Read($dbh)
    {
        try {

            $getLogs = $dbh->prepare("SELECT * FROM searchlogs");
            $getLogs->execute();
            $logArray = $getLogs->fetchAll(PDO::FETCH_ASSOC);
            return $logArray;

        } catch (PDOException $e)
        {
            $e->getMessage();
        }
    }

    public function Update($dbh)
    {
        // TODO: No need for admins to edit logs //
    }

    public function Delete($dbh)
    {
        $mID=$_GET["MemID"];
        try {
            $query = $dbh->prepare("DELETE FROM members WHERE MemID = $mID
LIMIT 1");
            $query->execute();
        } catch (PDOException $e) {
            $e->getMessage();
        }
    }
}

```

src >Staff.php

```

<?
require_once '../SCFC1/src/interfaces/IStaffInterface.php';
require_once "../SCFC1/src/models/ClubPersonnel.php";
require_once "../SCFC1/config/conninfo.php";

class staff implements IStaffRepository {

    public function Create($dbh, ClubPersonnel $staff)
    {
        try {

            $fName=$_POST["FirstName"];
            $sName=$_POST["Surname"];

```

```

        $dob=$_POST["DOB"];
        $position=$_POST["Position"];
        $sAddress=$_POST["StreetAddress"];
        $city=$_POST["City"];
        $pCode=$_POST["Postcode"];
        $eMail=$_POST["eMail"];
        $phone=$_POST["PhoneNo"];

        $insert = $dbh->prepare("INSERT INTO clubpersonnel(FirstName,
Surname, DOB, Position, StreetAddress, City, Postcode, eMail, PhoneNo)
VALUES (:FirstName, :Surname, :DOB, :Position, :StreetAddress, :City,
:Postcode, :eMail, :PhoneNo)");
        $insert->bindParam(':FirstName', $fName);
        $insert->bindParam(':Surname', $sName);
        $insert->bindParam('DOB', $dob);
        $insert->bindParam(':Position', $position);
        $insert->bindParam(':StreetAddress', $sAddress);
        $insert->bindParam(':City', $city);
        $insert->bindParam(':Postcode', $pCode);
        $insert->bindParam(':eMail', $eMail);
        $insert->bindParam(':PhoneNo', $phone);
        $insert->execute();
        return true;
    } catch (PDOException $e) {
        throw $e;
    }
}

public function Read($dbh)
{
    try {

        $getStaff = $dbh->prepare("SELECT * FROM clubpersonnel");
        $getStaff->execute();
        $staffArray = $getStaff->fetchAll(PDO::FETCH_ASSOC);
        return $staffArray;

    } catch (PDOException $e)
    {
        $e->getMessage();
    }
}

public function Update($dbh)
{
    try {

        $eNum=$_GET["EmployeeNo"];

        $fName=$_POST["FirstName"];
        $sName=$_POST["Surname"];
        $dob=$_POST["DOB"];
        $position=$_POST["Position"];
        $sAddress=$_POST["StreetAddress"];
        $city=$_POST["City"];
        $pCode=$_POST["Postcode"];
        $eMail=$_POST["eMail"];
        $phone=$_POST["PhoneNo"];

        $insert = $dbh->prepare("UPDATE clubpersonnel
SET FirstName = IF('$fName' = '', FirstName, '$fName'),
Surname = IF('$sName' = '', Surname, '$sName'),
DOB = IF('$dob' = '', DOB, '$dob'),
Position = IF('$position' = '', Position, '$position'),
StreetAddress = IF('$sAddress' = '', StreetAddress,

```

```

'$sAddress'),
    City = IF('$city' = '', City, '$city'),
    Postcode = IF('$pCode' = '', Postcode, '$pCode'),
    eMail = IF('$eMail' = '', eMail, '$eMail'),
    PhoneNo = IF('$phone' = '', PhoneNo, '$phone') WHERE EmployeeNo
= $eNum LIMIT 1");

    $insert->execute();
    return true;

} catch (PDOException $e) {
    throw $e;
}

}

public function Delete($dbh)
{
    $eNum=$_GET["EmployeeNo"];
    try {
        $query = $dbh->prepare("DELETE FROM clubpersonnel WHERE
EmployeeNo = $eNum LIMIT 1");
        $query->execute();
    } catch (PDOException $e) {
        $e->getMessage();
    }

}

}

}

?>

```

src >User.php

```

<?

require_once '../SCFC1/src/interfaces/INewUserObserver.php';
require_once '../SCFC1/src/interfaces/IObserver.php';
require_once "../SCFC1/config/conninfo.php";

class User implements INewUserObserver
{
    // Attributes

    // Properties

    // Methods

    // Register new Members
    public function Register($dbh, $fName, $sName, $eMail, $password)
    {
        try {

            $hashPassword = password_hash($password, PASSWORD_DEFAULT);

            $insert = $dbh->prepare("INSERT INTO members(FirstName, Surname,
eMail, password)
                                VALUES (:FirstName, :Surname, :eMail,
:password)");

            $insert->bindParam(':FirstName', $fName);
            $insert->bindParam(':Surname', $sName);
            $insert->bindParam(':eMail', $eMail);
            $insert->bindParam(':password', $hashPassword);
            $insert->execute();

```

```

        ?>
        <div class="news-success-msg">
            <p>
                <? echo "You have been successfully registered!";
                header('Refresh:1; url:../SCFC1/login.php');
            ?>
            </p>
        </div>
    <?
    }
    catch (PDOException $e) {
        throw $e;
    }

    $this->Notify($eMail);
}

public function RegisteredCheck($dbh, $eMail)
{
    try {
        $regCheck = $dbh->prepare("SELECT eMail FROM members WHERE
eMail=:eMail LIMIT 1");
        $regCheck->bindParam(':eMail', $eMail);
        $regCheck->execute();
        $clashedRow = $regCheck->fetch(PDO::FETCH_ASSOC);

        if($clashedRow["eMail"] == $eMail)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (PDOException $e) {
        echo $e->getMessage();
    }
}

// Log Users In
public function LogIn($dbh, $eMail, $password)
{
    try {
        // Get all users from members table
        $member = $dbh->prepare("SELECT * FROM members WHERE eMail=:eMail
LIMIT 1");
        $member->bindParam(":eMail", $eMail);
        $member->execute();

        // Returned row as array indexed by column name
        $result = $member->fetch(PDO::FETCH_ASSOC);

        // Check if row has been returned
        if ($member->rowCount() > 0)
        {
            if (password_verify($password, $result["password"]))
            {
                $_SESSION["userLoggedIn"] = $result["eMail"];
                return true;
            }
            else

```

```

        {
            return false;
            echo 'Incorrect Username or Password';
        }
    }

}

catch (PDOException $e)
{
    echo $e->getMessage();
}

}

// TODO: GetFirstName Something wrong with query? invalid argument for
foreach
// public function GetName($dbh)
// {
//     $mID = $_SESSION["userLoggedIn"];
//     try {
//         $getName = $dbh->prepare("SELECT * from members WHERE eMail =
$mID");
//         $getName->execute();
//         $nameMatch = $getName->fetchAll(PDO::FETCH_ASSOC);
//         return $nameMatch;
//     }
//     catch (PDOException $e) {
//         echo $e->getMessage();
//     }
// }

// Redirect Users
public function Redirect()
{
    header('Header: ../SCFC1/index.php');
}

// Check if User is Logged in
public function LoggedIn()
{
    if (isset($_SESSION["userLoggedIn"]) && $_SESSION["userLoggedIn"] !=
"")
    {
        return true;
    }
}

// function check_login () {
//     if(isset($_SESSION['login'] && $_SESSION['login'] != '') {
//         return true;
//     } else {
//         false;
//     }
// }

public function TimeOut()
{
    if (isset($_SESSION["LastActivity"]) && (time() -
$_SESSION["LastActivity"] > 300)) {
        session_start();
    }
}

```

```

        unset($_SESSION["userLoggedIn"]);
        session_destroy();
        echo "Logged out. Redirecting...";
        header ("Location: ../SCFC1/timeout.php");
        return true;
    }
    $_SESSION["LastActivity"] = time();
}

// Log Out
public function Logout()
{
    session_start();
    unset($_SESSION["userLoggedIn"]);
    session_destroy();
    echo "Logged out. Redirecting...";
    header( "refresh:2;url=../SCFC1/index.php" );
    return true;
}

// Observer Code

private $_newUser = array();

public function Attach($newUser)
{
    $this->_newUser[] = $newUser;
}

public function Detach($newUser)
{
    // TODO: Implement Detach() method.
}

public function Notify($eMail)
{
    foreach ($this->_newUser as $nUser)
    {
        $nUser->NewUserReg($this, $eMail);
    }
}
}
?>

```

admin.php

```

<? ob_start(); ?>
<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

    <? include "../SCFC1/layout/header.php"; ?>

    <?

```

```

///// SESSION TIMEOUT /////
$admin->TimeOut();

if(@$_SESSION["adminLoggedIn"] != "") {

    echo '<div class="container row">
        <h1 class="col-12 admin-log">Administrator Options</h1>
    </div>';

?>

<section>

    <div class="container row">
        <div class="col-12 but-container">
            <form action="" method="get">
                <button type="submit" name="option" id="submitButton2"
value="Staff">Staff</button>
                <button type="submit" name="option" id="submitButton2"
value="Players">Players</button>
                <button type="submit" name="option" id="submitButton2"
value="Clubs">Clubs </button>
                <button type="submit" name="option" id="submitButton2"
value="Fixtures">Fixtures</button>
                <button type="submit" name="option" id="submitButton2"
value="Admins">Admin</button>
                <button type="submit" name="option" id="submitButton2"
value="Members">Members</button>
                <button type="submit" name="option" id="submitButton2"
value="Logs">Search Logs</button>
            </form>
        </div>
    </div>

    <div class="">
        <div class="table-container">

            <?

switch(@$_REQUEST['option']) {

    case 'Staff': require
'../SCFC1/layout/admintables/stafftable.php';
        break;
    case 'Players': require
'../SCFC1/layout/admintables/playertable.php';
        break;
    case 'Clubs': require
'../SCFC1/layout/admintables/clubtable.php';
        break;
    case 'Fixtures': require
'../SCFC1/layout/admintables/matchtable.php';
        break;
    case 'Admins': require
'../SCFC1/layout/admintables/admintable.php';
        break;
    case 'Members': require
'../SCFC1/layout/admintables/membertable.php';
        break;
    case 'Logs': require '../SCFC1/layout/admintables/logtable.php';
        break;
}

```

```

        default: echo '<div class="spacer" style="height:
150px;"><p>Please Choose a Table from Above Options</p><div>';
    }

    ?>
        </div>
    </div>

</section>

<?
}
else {
    echo '<div class="container row">
        <h1 class="col-12 admin-log">Please Log In!</h1>

        <div class="col-12 big-login">
            <a href="adminlogin.php"></a>
        </div>
    </div>
    <div class="container row admin-img">
        
    </div>';
}

?>

<footer id="mainFooter">
    <div class="container row">

        <?
        switch(@ $_REQUEST['option']) {
            case 'Staff': require
'../SCFC1/layout/create/createstaff.php';
            break;
            case 'Players': require
'../SCFC1/layout/create/createplayers.php';
            break;
            case 'Clubs': require
'../SCFC1/layout/create/createclub.php';
            break;
            case 'Fixtures': require
'../SCFC1/layout/create/creatematch.php';
            break;
            case 'Admins': require
'../SCFC1/layout/create/createadmin.php';
            break;
            case 'Members': require
'../SCFC1/layout/create/createmember.php';
            break;
            default:
        }

        ?>

        <? include 'layout/sponsors.php'; ?>
    </footer>
    <? include "../SCFC1/scripts/scripts.php"; ?>

```

```
</body>
</html>
```

adminlogin.php

```
<?ob_start()?>
<!DOCTYPE html>
<html lang="en">
<head>
<? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

<? include "layout/header.php"; ?>

    <section id="ownerBanner">
        <h1 class="col-12 owners-caption">Admin Log In</h1>
    </section>

    <main>

        <div class="container row">
            <form class="form" action="" method="post">
                <fieldset>

                    <div class="personal">
                        <h4 class="col-12 admin-log">Administrator Log
In</h4>

                        <div class="col-6 entry-box4">
                            <input type="text" name="EmployeeNo"
placeholder="Employee Number">
                        </div>
                        <div class="col-6 entry-box5">
                            <input type="password" name="Password"
placeholder="Password">
                        </div>
                    </div>

                    <input type="submit" name="loginSubmit"
id="submitButton1" value="Log In">

                </fieldset>
            </form>

            <?
ob_start();
include "../SCFC1/config/conninfo.php";

//          if ($admin->LoggedIn() == true)
//          {
//              $admin->Redirect();
//          }
//          else{
//              // Do nothing
//          }

            if (isset($_POST["loginSubmit"])) {
```

```

        $loginSubmit = $_POST["loginSubmit"];
    }

    if (isset($loginSubmit)) {

        $eNumber=$_POST["EmployeeNo"];
        $password=$_POST["Password"];

        if($admin->Login($dbh, $eNumber, $password) == true)
        {
            //$_SESSION["adminLoggedIn"] = $eNumber;
            $admin->Redirect();
            // header ("Location: ../SCFC1/admin.php");

        }
        else
        {
            echo '<div class="login-mess">
                <p>Incorrect Username or Password</p>
            </div>';
        }

    }
    ob_end_flush();
?>

</div>

</main>

<footer id="mainFooter">
    <? include "layout/sponsors.php"; ?>
</footer>
<? include "../SCFC1/scripts/scripts.php"; ?>
</body>
</html>

```

delete.php

```

<? ob_start(); ?>
<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

    <? include "layout/header.php"; ?>

    <?

    //// SESSION TIMEOUT ////

    $admin->TimeOut();

    //////////////////////////////////

    if($_SESSION["adminLoggedIn"] != "") {

```

```

        echo '<div class="container row">
            <h1 id="hidden" class="col-12 admin-log">Administrator
Options</h1>
        </div>';

    ?>

    <section>

        <div class="container row">
            <div class="del-msg1">

                <? include "../SCFC1/config/conninfo.php";

                $table=$_GET["tableNo"];

                if ($table == 1) {

                    $personnel->Delete($dbh);
                    header("refresh:3; url=admin.php?option=Staff");

                }
                if ($table == 2) {

                    $players->Delete();
                    header("refresh:3; url=admin.php?option=Players");

                }
                if ($table == 3) {

                    $club->Delete();
                    header("refresh:5; url=admin.php?option=Clubs");

                }
                if ($table == 4) {

                    $match->Delete();
                    header("refresh:3; url=admin.php?option=Fixtures");

                }
                if ($table == 5) {

                    $admin->Delete();
                    header("refresh:3; url=admin.php?option=Admins");

                }
                if ($table == 6) {

                    $members->Delete();
                    header("refresh:3; url=admin.php?option=Mailing+List");

                }
                if ($table == 7) {

                    $log->Delete();
                    header("refresh:3; url=admin.php?option=Search+Log");

                }

                echo "<p>Record Successfully Deleted</P>";
                echo "<p class='redirect1'>You will be redirected shortly...</p>";

            ?>

        </div>
    </div>
    <div class="spacer" style="height: 325px;"></div>

```

```

</section>

<?
}
else {
    echo '<div class="container row">
        <h1 class="col-12 admin-log">Please Log In!</h1>

        <div class="col-12 big-login">
            <a href="adminlogin.php"></a>
        </div>
    </div>
    <div class="container row admin-img">
        
    </div>';
}

?>

<? include 'layout/sponsors.php'; ?>

<script type="text/javascript" src="js/scrollFuntions.js"></script>

</body>
</html>

```

edit.php

```

<? ob_start(); ?>
<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

    <? include "layout/header.php"; ?>

    <?

    //// SESSION TIMEOUT ////

    $admin->TimeOut();

    ///////////////////////////////////

    if(@$_SESSION["adminLoggedIn"] != "") {

        echo '<div class="container row">
            <h1 class="col-12 admin-log">Edit Record</h1>

```

```

        </div>';

?>

<section>
    <div class="del-msg">

        <? include "../SCFC1/config/conninfo.php";

        $table=$_GET["tableNo"];

        ////////// STAFF TABLE //////////

        require '../SCFC1/layout/edit/editstaff.php';

        ////////// PLAYER TABLE //////////

        require '../SCFC1/layout/edit/editplayer.php';

        ////////// CLUB TABLE //////////

        require '../SCFC1/layout/edit/editclub.php';

        ////////// FIXTURES TABLE //////////

        require '../SCFC1/layout/edit/editmatch.php';

        ////////// ADMIN TABLE //////////

        require '../SCFC1/layout/edit/editadmin.php';

        ////////// MEMBERS //////////

        require '../SCFC1/layout/edit/editmember.php';

?>

    </div>

</section>

<?

}
else {

    echo '<div class="container row">
        <h1 class="col-12 admin-log">Please Log In!</h1>

        <div class="col-12 big-login">
            <a href="adminlogin.php"></a>
        </div>
    </div>
    <div class="container row admin-img">
        
    </div>';

}

?>

```

```

        <footer id="mainFooter">
            <div class="container row">
                <? include 'layout/sponsors.php'; ?>
            </div>
        </footer>
        <script type="text/javascript" src="js/scrollFuntions.js"></script>

</body>
</html>

```

index.php

```

<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

    <? include "layout/header.php"; ?>

    <section id="banner">
        <h1 class="col-12 caption">
            <a href="news.php">Courtney can't wait to get<br>
                back to the liberty</a>
        </h1>
    </section>

    <footer id="mainFooter">
        <? include "layout/sponsors.php"; ?>
    </footer>
    <? include "../SCFC1/scripts/scripts.php"; ?>

</body>
</html>

```

login.php

```

<? ob_start() ?>
<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
<button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

<? include "layout/header.php"; ?>

```

```

<section id="logInBanner">
    <h1 class="col-12 login-caption">Jack Army Log In</h1>
</section>

<main>

    <div class="container row">
        <form class="form" action="#submitButton1" method="post">
            <fieldset>

                <div class="personal">
                    <h4 class="col-12 admin-log">Member Log In</h4>
                    <div class="col-6 entry-box4">
                        <input type="text" name="eMail" placeholder="e-mail
address">
                    </div>
                    <div class="col-6 entry-box5">
                        <input type="password" name="Password"
placeholder="Password">
                    </div>
                </div>

                <input type="submit" name="loginSubmit" id="submitButton1"
value="Log In">

            </fieldset>
        </form>

        <?

require_once "../SCFC1/config/conninfo.php";

///// SESSION TIMEOUT /////
$user->TimeOut();

// Is User Logged In
//     if ($user->LoggedIn() == true)
//     {
//         $user->Redirect('../SCFC1/index.php');
//     }
//     else
//     {
//         // Do nothing
//     }

if (isset($_POST["loginSubmit"])) {

    $loginSubmit = $_POST["loginSubmit"];

}

if (isset($loginSubmit)) {

    $eMail=$_POST["eMail"];
    $password=$_POST["Password"];

    if ($user->Login($dbh, $eMail, $password) == true)
    {

        header('Location: ../SCFC1/index.php');
        // TODO: Get thie redirect working by passing in variables.
        //$user->Redirect('../SCFC1/index.php');
    }
    else
    {

```

```

        ?>
        <div class="login-mess">
            <p>
                <? echo "Incorrect Username or Password"; ?>
            </p>
        </div>
        <?
    }
}

?>

</div>

</main>

<footer id="mainFooter">
    <? include "layout/sponsors.php"; ?>
</footer>
<? include "../SCFC1/scripts/scripts.php"; ?>

</body>
</html>

```

logout.php

```

<?

require_once "../SCFC1/config/conninfo.php";

$user->Logout();
//header('Location: ../SCFC1/index.php');

?>

```

match.php

```

<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

    <?

    include "layout/header.php";
    require_once "../SCFC1/config/conninfo.php";

    if ($user->LoggedIn() || $admin->LoggedIn())
    {
        ?>

    <section id="matchBanner"></section>

```



```

                                <? echo "<a
href=\"match.php?page=$prevs#matches\"><img src='../SCFC1/images/black-left-
round-48.png'</a>"; ?>
                                </div>
                                <?>

                                $pages = intval ($fixture->GetFixtures($dbh) / $limit);

                                if ($fixture->GetFixtures($dbh) % $limit)
                                {
                                    $pages++;
                                }

                                if (!((( $page + $limit) / $limit) == $pages) && $pages
!= 1)
                                {
                                    $news = $page + $limit; ?>
                                    <div id="matches" class="col-6 arrows-r">
                                        <? echo "<a
href=\"match.php?page=$news#matches\"><img src='../SCFC1/images/black-right-
round-48.png'</a>"; ?>
                                        </div>
                                        <?>
                                    }
                                ?>
                                </div>
                            </div>
                        </main>
                        <?
                    }
                    else {
                        include '../SCFC1/layout/please-log-in.php';
                    }
                ?>

                <footer id="mainFooter">
                    <? include "layout/sponsors.php"; ?>
                </footer>
                <? include "../SCFC1/scripts/scripts.php"; ?>
            </body>
        </html>

```

news.php

```

<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

    <? include "layout/header.php"; ?>

    <section id="newsBanner">
        <h1 class="col-12 news-caption">
            <a href="news.php">Courtney can't wait to get<br>
                back to the liberty</a>
        </h1>
    </section>

```

```

<main>

  <div class="container row">
    <div id="articleCon">
      <div class="col-9">
        <p class="heading">COURTNEY BAKER-RICHARDSON SAYS SWANSEA CITY
ARE CHOMPING AT THE BIT TO GET BACK ON HOME SOIL WHEN THEY HOST SHEFFIELD
UNITED AT THE LIBERTY STADIUM ON SATURDAY (5.30PM)</p>
        <br/>
        <br/>
        <p class="words">The Swans have yet to play a home fixture in
2019, having travelled to Reading and Preston in the Championship, as well
as facing an away draw against Aston Villa in the FA Cup.
        <br/>
        <br/>
        Graham Potter's men have come through that trio of fixtures
undefeated, and can now look forward to having four of their next five games
in front of their own fans.
        <br/>
        <br/>
        "The Liberty Stadium is a fortress and we love it when we play
there," said Baker-Richardson, who has scored in his last two appearances.
        <br/>
        <br/>
        "Our fans are a credit to us when they are packing out the
Liberty, or so many of them are travelling to our away games.

        <div class="article-image"></div>

        <div>
          <p class="words1">"It is a great atmosphere whenever we step in
there and play, and we need to bring the frustration we felt at Preston and
channel that energy into making sure we win there.
          <br/>
          <br/>
          "We have our next three games at home and if we win those we can
be in and around it".
          <br/>
          <br/>
          "Points are points and we need them."
        </div>

      </div>
    </div>
  </div>

</main>

<footer id="mainFooter">
  <? include "layout/sponsors.php"; ?>
</footer>
<? include "../SCFC1/scripts/scripts.php"; ?>

</body>
</html>

```

register.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
<button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

<? include "layout/header.php"; ?>

<section id="signUpBanner">
    <h1 class="col-12 register-caption">Register</h1>
</section>

<main>

</main>

<footer id="mainFooter">
    <? include "layout/signup.php"; ?>
</footer>
<? include "../SCFC1/scripts/scripts.php"; ?>

</body>
</html>
```

stadium.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>

<body>
    <button onclick="backToTop()" id="topButton" title="Back To
    Top">&#708;</button>

    <? include "layout/header.php"; ?>

    <section id="bannerCon">
        <h1 class="col-12 caption-lib">Liberty Stadium</h1>
    </section>

    <main>

        <div class="container row">

            <div class="col-6 ct-info">
                <p>Liberty Stadium</p>
                <p>
                    Landore
                    <br />
                    Swansea
                    <br />
                </p>
            </div>
        </div>
    </main>
</body>
</html>
```

```

SA1 2FA
<br />
Wales, UK.
</p>
<p>+448172 223 224</p>
<a
href="mailto:info@libertystadium.com">info@libertystadium.com</a>

</div>

<div class="col-6 map">
    <iframe width="600" height="450" frameborder="0"
style="border:0"
src="https://www.google.com/maps/embed/v1/place?q=place_id:ChIJxfdmomj1bkgRN
-XgukaNFx4&key=AIzaSyCGlR_D1ioDk5YnZirS4PSV_l8ofboT1pw"
allowfullscreen></iframe>
</div>

</div>

</main>

<footer id="mainFooter">
    <? include "layout/sponsors.php"; ?>
</footer>
<? include "../SCFC1/scripts/scripts.php"; ?>

</body>
</html>

```

team.php

```

<!DOCTYPE html>
<html lang="en">
<head>
    <? include '../SCFC1/head/headdata.php'; ?>
</head>
<body>
    <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>

    <?
include "layout/header.php";
if($user->LoggedIn() || $admin->LoggedIn())
{
    ?>
<section id="teamBanner"></section>
<main>
    <div class="container row">
        <h4 class="col-12 fix-res">Swansea City AFC - The Team</h4>
    </div>
    <div class="container row">
        <p class="s-words">Search Swansea City Players Using the Search
Box Below...</p>
    </div>
    <div class="container row">
        <form class="form" action="#submitButton1" method="post">
            <fieldset>
                <div class="personal">
                    <div class="col-6 entry-box4">
                        <input type="text" name="search"
placeholder="Enter Search">

```

```

        </div>
        <p class="col-6 s-by">Search by...</p>
        <div class="col-5 selections">
            <input type="radio" name="option"
value="Surname" checked="checked"><label>Surname</label>
            <input type="radio" name="option"
value="FirstName"><label>First Name</label>
            <input type="radio" name="option"
value="PlayingPosition"><label>Player Position</label>
            <input type="radio" name="option"
value="PlayerNo"><label>Player Number</label>
        </div>
        </div>
        <input type="submit" name="searchSubmit"
id="submitButton1" value="Search">
    </fieldset>
</form>
<?
include "../SCFC1/config/conninfo.php";
if (isset($_POST["searchSubmit"])) {$searchSubmit =
$_POST["searchSubmit"];}
if (isset($searchSubmit)) {

    $keyword=$_POST["search"];
    $filter=$_POST["option"];
    $userEmail = $_SESSION["userLoggedIn"];
    $dateTime = date('Y-m-d H:i:s');

    $data = $logSearch->Search($dbh, $filter, $keyword);
    $log = $logSearch->SearchLog($dbh, $userEmail, $keyword,
$filter, $dateTime);

    foreach($data as $row)
    {?>
        <div class="col-12 php-res-player">
            <div class="php-res-name">
                <p><?
                    echo $row["FirstName"]." ";
                    echo $row["Surname"]." ";
                ?>
            </p>
            </div>
            <div class="php-res-img">
                <? $pic=$row["Photo"];
                if($pic=="") //If picture isn't there
                {?><?}
                else // Otherwise show the image in the database
                {?>" width="200" height="250"><?}?>
            </div>
            <div class="php-res-other">
                <p>
                    <?
                    echo "Position:
".$row["PlayingPosition"]."<br>";
                    echo "Number: ".$row["PlayerNo"]."<br>";
                    $dob = new DateTime($row["DOB"]);
                    $now = new DateTime('today');
                    echo "Age: ".$dob -> diff($now) ->y;
                    //echo "D.O.B: ".$row["DOB"]."<br>";
                ?>
            </p>
            </div>
        </div>
    </div>
}

```

```

        }
    }
    ?>
</div>
</main>
<?
} else {
    include '../SCFC1/layout/please-log-in.php';
}
?>

<footer id="mainFooter">
    <? include "layout/sponsors.php"; ?>
</footer>
    <? include "../SCFC1/scripts/scripts.php"; ?>
</body>
</html>

```

timeout.php

```

<!DOCTYPE html>
<html lang="en">
    <head>
        <? include '../SCFC1/head/headdata.php'; ?>
    </head>
        <? include "../SCFC1/layout/header.php"; ?>
    <body>
        <button onclick="backToTop()" id="topButton" title="Back To
Top">&#708;</button>
        <section>
            <div class="container row">
                <div class="sess-mess">
                    <p>Session Timed Out! Please Log In Again</p>
                </div>
            </div>
        </section>
        <footer id="mainFooter">
            <div class="container row">
                <? include 'layout/sponsors.php'; ?>
            </div>
        </footer>
        <script type="text/javascript" src="js/scrollFuntions.js"></script>
    </body>
</html>

```